

# z/OS Version 1 Release 9 Implementation

JES2, JES3, GRS, SMF, WLM, z/OS  
UNIX, zFS

Health Checker, SDSF, System  
REXX, Binder, DFSMS

Message flood, XML, CIM,  
z/OS base



Paul Rogers - Paul-Robert Hering  
Lutz Kuehner - Jean-Louis Lafitte  
Marcos Minatto - Jaqueline Mourao  
Meganen Naidoo - Gil Peleg  
Evanir Philipi - Giancarlo Rodolfi

**Redbooks**





International Technical Support Organization

**z/OS Version 1 Release 9 Implementation**

December 2007

Archived

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

Archived

**First Edition (December 2007)**

This edition applies to Version 1 Release 9 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2007. 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

<b>Notices</b> .....	xiii
Trademarks .....	xiv
<b>Preface</b> .....	xv
The team that wrote this book .....	xvi
Become a published author .....	xvii
Comments welcome .....	xvii
<b>Chapter 1. z/OS Version 1 Release 9</b> .....	1
1.1 z/OS V1R9 enhancements .....	2
1.2 BCP miscellaneous enhancements .....	4
1.3 z/OS support for IBM System z servers .....	4
1.4 z/OS UNIX System Services .....	4
1.5 z/OS Workload Manager .....	6
1.6 Console message flood .....	7
1.7 System Logger .....	7
1.8 SMF use of System Logger .....	7
1.9 Coupling Facility enhancements .....	7
1.10 GRS 64-bit exploitation .....	8
1.11 Sysplex failure management .....	9
1.12 Program management binder .....	9
1.13 XCF Couple Data Set .....	10
1.14 Language Environment .....	10
1.15 DFSMS enhancements .....	11
1.16 z/OS Communications Server .....	13
1.17 z/OS security .....	17
1.18 Spool Display .....	18
1.19 System REXX .....	18
1.20 IBM Health Checker for z/OS .....	18
1.21 Alternate Library for REXX .....	19
1.22 RRS .....	19
1.23 ISPF .....	19
1.24 Common Information Model .....	20
1.25 Metal C runtime library .....	20
1.26 XML System Services .....	21
1.27 z/OS dbx enhancements .....	21
1.28 Unicode .....	21
<b>Chapter 2. Installation considerations</b> .....	23
2.1 Ordering z/OS V1R9 .....	24
2.1.1 Hardware requirements .....	24
2.1.2 Export control features .....	24
2.2 New base elements .....	24
2.2.1 Alternate Library for REXX .....	25
2.2.2 Metal C Runtime Library .....	25
2.2.3 Elements changed in z/OS V1R9 .....	25
2.3 Functions withdrawn from z/OS V1R9 .....	25
2.4 Functions withdrawn in a future release .....	26
2.4.1 Changes to driving system requirements .....	26

2.5	Changed base elements and optional features . . . . .	27
2.6	Coexistence, fallback, and migration . . . . .	28
2.7	54-way support with the z9 EC . . . . .	29
2.8	New address spaces . . . . .	29
2.9	System z New Application License Charges (zNALC) . . . . .	30
2.9.1	zNALC support . . . . .	30
2.9.2	NALC users . . . . .	33
2.9.3	zNALC and SCRT and APAR OA20314 . . . . .	34
<b>Chapter 3. Coupling Facility enhancements . . . . .</b>		<b>35</b>
3.1	CF duplexing performance enhancements . . . . .	36
3.1.1	CFLEVEL 15 . . . . .	37
3.2	CF measurement enhancements . . . . .	39
3.2.1	RMF enhancements . . . . .	40
3.3	RMF Monitor III Data Portal for z/OS . . . . .	42
3.4	CF maintenance mode . . . . .	46
3.4.1	Migration and coexistence . . . . .	47
3.4.2	Using the CF maintenance mode . . . . .	47
<b>Chapter 4. ICSF support for PKCS #11 . . . . .</b>		<b>53</b>
4.1	PKCS #11 overview . . . . .	54
4.2	z/OS ICSF overview . . . . .	54
4.3	ICSF: PKCS #11 support . . . . .	55
4.3.1	PKCS #11 integration into z/OS . . . . .	55
4.3.2	Updating your ICSF definition to support PKCS #11 . . . . .	56
4.3.3	RACF and z/OS PKCS #11 token services . . . . .	57
4.3.4	Migration considerations . . . . .	59
4.4	Using PKCS11 token browser utility panels . . . . .	60
4.4.1	Running ICSF in a sysplex environment . . . . .	61
<b>Chapter 5. Allocation dynamic storage improvements . . . . .</b>		<b>63</b>
5.1	Overview . . . . .	64
5.2	Allocation improvements in z/OS V1R9 . . . . .	65
<b>Chapter 6. System Logger enhancements . . . . .</b>		<b>67</b>
6.1	System Logger overview . . . . .	68
6.1.1	Log stream exploiters . . . . .	70
6.1.2	z/OS V1R8 improvements of log stream data sets recall . . . . .	71
6.2	z/OS V1R9 improvements of log stream data set recalls . . . . .	72
6.3	Cleanup of CF list entries for unconnected log streams . . . . .	74
6.4	System Logger publication updates . . . . .	74
<b>Chapter 7. SMF recording to log streams . . . . .</b>		<b>75</b>
7.1	SMF overview . . . . .	76
7.1.1	SMF and log streams with z/OS V1R9 . . . . .	77
7.2	Installation of SMF log streams . . . . .	79
7.2.1	Defining SMF log streams . . . . .	79
7.2.2	Updating the CFRM policy for SMF CF structure logstream . . . . .	82
7.2.3	Updating the SMFPRMxx parmlib member . . . . .	83
7.2.4	SMFPRMxx parmlib member considerations . . . . .	85
7.2.5	Switching to log stream mode . . . . .	86
7.3	Dumping the SMF log stream data set . . . . .	88
7.3.1	Using the SWITCH command with log streams . . . . .	91

7.4 Migration considerations . . . . .	91
<b>Chapter 8. GRS enhancements . . . . .</b>	<b>93</b>
8.1 Global resource serialization overview . . . . .	94
8.1.1 Setting address space ENQ limits . . . . .	94
8.1.2 Contention notification system movement . . . . .	97
8.2 GRS storage constraint relief with z/OS V1R9 . . . . .	99
8.2.1 Ensure that GRSCNFxx is used properly for GRS=NONE . . . . .	101
8.2.2 GRS exit routines in cross-memory mode . . . . .	101
8.2.3 ISGADMIN enhancement . . . . .	102
8.3 GRS performance enhancements with z/OS V1R9 . . . . .	103
8.4 GRS debugging improvements . . . . .	103
<b>Chapter 9. Message Flood Automation . . . . .</b>	<b>105</b>
9.1 Message Flood Automation overview . . . . .	106
9.2 Message Flood Automation implementation . . . . .	106
9.2.1 Message flood problems . . . . .	107
9.2.2 MPF processing . . . . .	107
9.2.3 MPF processing exit . . . . .	108
9.3 Installing Message Flood Automation . . . . .	109
9.3.1 Message Flood Automation exits . . . . .	109
9.3.2 Loading and activating . . . . .	113
9.4 Customization and tuning . . . . .	115
9.4.1 Providing a MSGFLDxx parmlib member . . . . .	115
9.4.2 Types of message classes processed . . . . .	116
9.4.3 Message class controls . . . . .	117
9.4.4 Message Flood Automation guidelines . . . . .	120
9.4.5 Turning Message Flood Automation ON or OFF . . . . .	122
9.4.6 Displaying your policy . . . . .	123
9.5 Command summary . . . . .	124
<b>Chapter 10. WLM enhancements . . . . .</b>	<b>127</b>
10.1 Promote jobs which have been cancelled . . . . .	128
10.1.1 z/OS V1R9 enhancement . . . . .	128
10.1.2 Migration and coexistence considerations . . . . .	128
10.2 Start a minimum number of servers . . . . .	129
10.2.1 z/OS V1R9 enhancement . . . . .	129
10.2.2 Exploiters of the new service request . . . . .	131
10.3 WLM enhancements for blocked workloads . . . . .	131
10.3.1 Promote higher dispatch priority . . . . .	132
10.4 RMF enhancements for blocked workloads . . . . .	133
10.4.1 RMF CPU Activity report . . . . .	133
10.4.2 RMF Workload Activity report . . . . .	134
10.4.3 New SMF record types . . . . .	135
10.4.4 RMF Distributed Data Server . . . . .	136
10.5 Improved assist processor routing services . . . . .	136
10.5.1 Sysplex routing services IWMSRSRS improvements . . . . .	137
10.5.2 Sysplex routing services IWM4SRSC improvements . . . . .	141
10.5.3 IWMWSYSQ service . . . . .	143
10.5.4 Migration and coexistence considerations . . . . .	144
10.6 Group capacity limit . . . . .	144
10.6.1 Defined capacity review . . . . .	144
10.6.2 Group capacity definition rules . . . . .	146
10.6.3 Group capacity example . . . . .	147

10.6.4	Hardware and software for group capacity . . . . .	147
10.6.5	Group capacity limit example . . . . .	149
10.6.6	RMF and SMF updates to support group capacity limit . . . . .	150
10.6.7	Examples related to usage of group capacity limit . . . . .	150
<b>Chapter 11.</b>	<b>C/C ++ enhancements . . . . .</b>	<b>155</b>
11.1	SUSv3 implementation in z/OS V1R9 . . . . .	156
11.1.1	z/OS V1R9 and SUSv3 . . . . .	156
11.1.2	Compiling an SUSv3 application . . . . .	157
11.1.3	Invoking Threads support . . . . .	158
11.1.4	Setting environment variables affects run-time behavior . . . . .	159
11.1.5	New APIs . . . . .	160
11.1.6	New Threading interfaces . . . . .	161
11.1.7	Modified APIs . . . . .	162
11.1.8	Migration and coexistence considerations . . . . .	162
<b>Chapter 12.</b>	<b>ISPF enhancements . . . . .</b>	<b>163</b>
12.1	Edit and browse z/OS UNIX files . . . . .	164
12.1.1	ISPF enhancement in z/OS V1R9 . . . . .	164
12.2	ISPF personal data set lists . . . . .	168
12.3	EDIT primary commands support . . . . .	169
12.4	EDIT macro command support . . . . .	173
12.5	ISPF services support . . . . .	174
12.6	PDF installation-wide data set allocation exit . . . . .	176
12.7	Support for editing ASCII data . . . . .	176
12.8	Mixed case in ISPF command tables . . . . .	178
<b>Chapter 13.</b>	<b>Security enhancements . . . . .</b>	<b>181</b>
13.1	RACF enhancements . . . . .	182
13.1.1	Password phrase minimum length change . . . . .	182
13.1.2	Writable key ring functions . . . . .	183
13.1.3	UTF8 characters support in digital certificates . . . . .	185
13.1.4	REFRESH warning message after RACDCERT commands . . . . .	185
13.2	Java security API . . . . .	186
13.3	System SSL enhancements . . . . .	189
13.3.1	Introduction to the SSL protocol . . . . .	189
13.3.2	Certificate revocation lists (CRLs) granularity . . . . .	191
13.3.3	Rehandshake notification . . . . .	193
13.3.4	Host name validation . . . . .	194
13.3.5	Hardware-to-software switch notification . . . . .	195
13.4	PKI Services enhancements . . . . .	196
13.4.1	Automatic certificate renewal processing . . . . .	196
13.4.2	RACF-style distinguished name . . . . .	198
13.4.3	E-mail notification for administrators . . . . .	198
13.4.4	Longer validity period for certificates . . . . .	199
13.4.5	Query on expiring certificates . . . . .	199
<b>Chapter 14.</b>	<b>z/OS Communication Server . . . . .</b>	<b>201</b>
14.1	zIIP-assisted IPsec . . . . .	202
14.1.1	Implementation of zIIP-assisted IPSEC . . . . .	202
14.1.2	Example of zIIP-assisted IPsec implementation . . . . .	203
14.2	Policy-based routing . . . . .	212
14.2.1	Policy-based routing implementation . . . . .	214
14.2.2	Policy-based routing implementation example . . . . .	217

<b>Chapter 15. System REXX for z/OS</b> .....	237
15.1 Introduction to System REXX (SYSREXX) .....	238
15.2 SYSREXX address space (AXR) .....	238
15.2.1 SYSREXX from consoles .....	239
15.2.2 AXREXX macro service .....	239
15.3 Customizing System REXX .....	241
15.4 Using System REXX .....	242
15.5 Usage and migration considerations .....	245
15.5.1 Writing REXX execs .....	245
15.5.2 Using input and output files .....	246
15.5.3 Other AXREXX parameters .....	247
15.5.4 Arguments and variables within a REXX exec .....	247
<b>Chapter 16. z/OS XL C/C++ Metal option</b> .....	251
16.1 Metal option introduction .....	252
16.1.1 XL C Metal compiler option .....	252
16.2 XL C Metal option .....	253
16.2.1 Metal option overview .....	254
16.2.2 Using the Metal option .....	254
16.2.3 Linkage conventions .....	256
16.2.4 AR-mode and the Metal option .....	256
16.2.5 Metal C runtime library .....	257
16.3 Decimal floating point .....	258
16.3.1 The need for decimal arithmetic .....	258
16.3.2 Extended precision floating-point numbers .....	258
16.3.3 New floating-point data types .....	259
16.3.4 Decimal arithmetic context .....	259
16.3.5 XL C/C++ support for decimal floating point data types .....	260
16.3.6 XL C/C++ run-time library .....	261
16.3.7 UNIX System Services dbx debugger .....	261
16.4 dbx support of WebSphere remote debuggers .....	262
16.5 Specialized hardware instructions support .....	266
16.5.1 Available new built-in functions .....	267
16.6 Migration considerations .....	268
16.7 Prelnit tracing .....	268
16.7.1 Migration considerations .....	269
16.7.2 Prelnit tracing characteristics .....	269
16.8 DLL diagnostics .....	272
16.8.1 Language Environment IPCS support .....	274
<b>Chapter 17. z/OS UNIX System Services</b> .....	275
17.1 Automove consistency .....	276
17.1.1 Problems with sysplex-aware file systems without the new support .....	277
17.1.2 New automove enhancements .....	278
17.1.3 Migration and coexistence considerations .....	282
17.2 zFS small enhancements .....	283
17.2.1 IOEAGFMT and IOEAGSLV authorization .....	283
17.2.2 Concurrent log recovery .....	286
17.2.3 Improved dynamic grow .....	286
17.2.4 Improved hang detection .....	287
17.2.5 Hang detection messages .....	287
17.2.6 Analyzing hang conditions .....	287
17.2.7 z/OS V1R9 enhancements .....	289

<b>Chapter 18. SDSF enhancements</b> .....	291
18.1 SDSF and the REXX programming language .....	292
18.1.1 SDSF REXX and System REXX .....	292
18.1.2 Authorization for SDSF and REXX .....	292
18.2 Setting up the SDSF host command environment .....	293
18.2.1 Issuing SDSF commands in a REXX program .....	294
18.2.2 Special REXX variables .....	295
18.3 Examples of using ISFEXEC .....	296
18.3.1 The WHO and QUERY commands .....	297
18.3.2 Issuing operator commands .....	298
18.3.3 Issuing action characters .....	300
18.3.4 Browsing job output .....	301
18.3.5 Printing job output .....	303
18.4 Executing REXX execs .....	304
18.4.1 Diagnosing errors in an SDSF REXX exec .....	304
18.5 SDSF migration considerations .....	305
<b>Chapter 19. New faces of z/OS</b> .....	307
19.1 Introduction to the new face of z/OS .....	308
19.1.1 z/OS ease of use enhancements .....	308
19.2 z/OS V1R9 and new faces of z/OS .....	309
19.2.1 System REXX .....	310
19.2.2 SDSF REXX .....	311
19.2.3 Using REXX to write health check routines .....	311
19.2.4 XL C Metal compiler option .....	311
19.2.5 Common event adapter .....	312
19.3 Common Information Model .....	312
19.3.1 z/OS V1R9 enhancements for CIM .....	313
19.3.2 CIM cross-platform management .....	314
19.3.3 CIM components and dependencies .....	315
19.4 CIM server overview .....	318
19.4.1 CIM server support in z/OS V1R9 .....	318
19.5 CIM client-to-CIM server access .....	321
19.6 CIM server runtime update and enhancements .....	323
19.6.1 Automatic Restart Manager support .....	323
19.6.2 SSL certificate-based authentication .....	324
19.6.3 Logging facility changed to syslog daemon .....	326
19.6.4 New command-line utility: cimsub .....	326
19.7 CIM client API for Java .....	327
19.8 Instrumentation in z/OS V1R9 .....	328
19.8.1 Required parmlib updates .....	330
19.8.2 Instrumentation for logical disk volumes .....	331
19.8.3 Instrumentation of batch jobs .....	331
19.8.4 Instrumentation for a sysplex .....	331
19.8.5 DFSMSrmm CIM provider .....	335
19.8.6 RMF CIM monitoring .....	335
19.9 Migration and coexistence considerations .....	338
19.9.1 General migration considerations .....	338
19.9.2 Cloning considerations .....	339
<b>Chapter 20. Program management enhancements</b> .....	341
20.1 New Binder options .....	342
20.1.1 The MODMAP Binder option .....	342
20.1.2 The INFO Binder option .....	343

20.2	Enhanced Binder control statements	343
20.3	SYSLMOD record format verification	344
20.4	Binder C/C++ API	345
20.5	Support for side deck definition files in archive files	347
20.6	Binder fast data access enhancements	348
<b>Chapter 21.</b>	<b>JES2 and JES3 enhancements</b>	<b>351</b>
21.1	JES2 enhancements	352
21.2	SSI requests authorization enhancements	352
21.2.1	SSI 11 - User destination validation/conversion service	353
21.2.2	SSI 70 - Scheduler facilities function	353
21.2.3	SSI 71 - JES job information services	353
21.2.4	SSI 75 - Notify user message service call	355
21.2.5	SSI 79 - SYSOUT application programming interface (SAPI)	355
21.2.6	SSI 80 - Extended status function call	355
21.2.7	SVC 99 - spool browse	356
21.3	\$C Job command enhancements	356
21.4	\$TRACE facility enhancements	357
21.4.1	TRACE initialization statement and \$T TRACE command	357
21.4.2	INTRDR tracing	359
21.5	Changes to JES2 exits	361
21.5.1	\$JCT eye catcher	361
21.5.2	Exit 8 - User environment \$CBIO	361
21.5.3	Exit 31 - Allocation SSI	361
21.5.4	Exit 42 and exit 45	361
21.6	JES3 enhancements	362
21.6.1	Relief of the OSE buffer number limit	362
21.6.2	Coexistence considerations	362
21.6.3	More efficient use of spool space	363
<b>Chapter 22.</b>	<b>IBM Health Checker for z/OS</b>	<b>365</b>
22.1	System REXX check support	366
22.2	Defining a REXX check	366
22.2.1	REXX check structure	368
22.2.2	DEBUG mode	369
22.2.3	Scheduling a REXX check	369
22.2.4	DELETE FORCE=YES	369
22.2.5	Procedure to implement a REXX check	370
22.3	Extended SDSF CK support	371
22.4	New checks available with z/OS V1R9	372
<b>Chapter 23.</b>	<b>DFSMS enhancements</b>	<b>375</b>
23.1	Basic Access Methods (BAM) performance	376
23.1.1	Long-term page fixing for BSAM data buffers	377
23.1.2	BSAM and QSAM support for MULTACC	377
23.1.3	QSAM support for MULTSDN	378
23.2	VSAM system managed buffering (SMB) enhancements	379
23.2.1	SMB overview	379
23.2.2	Installation considerations	380
23.3	Multi-volume data set in the same storage facility image	381
23.3.1	Storage facility image (SFI) overview	382
23.3.2	DFSMS volume selection enhancement	382
23.3.3	Storage Facility Image (SFI) attributes	382
23.3.4	DFSMS volume selection with SFI attribute	383

23.3.5 Migration considerations . . . . .	384
23.4 Object access method (OAM) enhancements . . . . .	384
23.4.1 Using OAM enhancements . . . . .	384
23.4.2 Miscellaneous enhancements . . . . .	387
23.4.3 Migration considerations . . . . .	388
23.5 DFSMSShsm enhancements . . . . .	388
23.5.1 Abend 878 reduction . . . . .	389
23.5.2 Functional statistics record (FSR) improvements . . . . .	390
23.5.3 Return priority (RP) exit ARCRPEXT changes . . . . .	392
23.6 DFSMSrmm enhancements . . . . .	393
23.6.1 Task management support . . . . .	393
23.6.2 Multitasking of utilities . . . . .	398
23.6.3 Control data set (CDS) serialization . . . . .	406
23.6.4 Migration and coexistence considerations . . . . .	409
23.6.5 Common Information Model (CIM) provider . . . . .	409
23.6.6 JCL data set names . . . . .	414
23.6.7 Data set names in RMM subcommands . . . . .	416
23.6.8 Shared parmlib support . . . . .	420
23.6.9 TSO subcommands . . . . .	422
23.6.10 3592 Model E05 software support . . . . .	430
23.6.11 Migration and coexistence considerations . . . . .	431
23.7 Network File Systems (NFS) enhancements . . . . .	432
23.7.1 24-bit addressing relief . . . . .	432
23.7.2 Multi TCP/IP stack support . . . . .	432
23.7.3 Usage and invocation . . . . .	433
23.7.4 AddDS operator command . . . . .	433
23.7.5 RACF data labeling . . . . .	434
23.7.6 NFS v4 client support . . . . .	435
23.7.7 Client Attribute syntax . . . . .	436
23.7.8 Server Ctrace upgrade . . . . .	436
23.7.9 Terminal ID based restricted MVSLOGIN . . . . .	438
<b>Chapter 24. Large format data sets . . . . .</b>	<b>439</b>
24.1 Large format data set overview . . . . .	440
24.2 TSO/E and large format data sets . . . . .	441
24.3 TSO PRINTDS command . . . . .	442
24.4 REXX and CLIST LISTDSI function . . . . .	442
24.5 Enhanced I/O capability in TSO/E for CLIST and REXX . . . . .	443
24.6 Messages related to new support . . . . .	443
24.7 Migration and coexistence considerations . . . . .	444
<b>Chapter 25. RMF enhancements . . . . .</b>	<b>445</b>
25.1 RMF enhancements for FICON . . . . .	446
25.1.1 SMF record changes . . . . .	447
25.2 RMF Monitor III Data Portal . . . . .	447
25.2.1 Sort capability for full Monitor III reports . . . . .	449
25.3 SpreadSheet Reporter enhancements . . . . .	451
25.3.1 New RMF Spreadsheet options . . . . .	451
25.3.2 zAAP and zIIP support . . . . .	452
25.3.3 Report Class periods . . . . .	454
25.3.4 RMF XCF Activity Report . . . . .	455
25.3.5 Process user-defined overview records . . . . .	457



<b>Chapter 26. XML enhancements</b> .....	459
26.1 XML System Services .....	460
26.2 Performance improvements .....	460
26.3 C/C++ APIs .....	461
26.3.1 Sample project .....	462
26.3.2 How to compile .....	464
26.3.3 zAAP considerations .....	465
<b>Chapter 27. RRS enhancements</b> .....	467
27.1 ATRQSRV batch support .....	468
27.1.1 ATRQSRV utility .....	468
27.2 Resource manager unregister .....	470
<b>Chapter 28. Language Environment</b> .....	475
28.1 iconv() enhancements .....	476
28.1.1 Migration actions .....	477
28.2 CEEDUMP enhancement .....	478
28.2.1 Enhanced traceback section .....	479
28.3 edcmtext utility .....	481
28.4 HEAPPOOLS performance improvement .....	482
28.5 z/OS UNIX support for ceebltdx utility .....	483
28.6 CLER run-time option change support .....	485
28.7 New and modified callable services .....	485
28.8 CEE3DLY and CEEDLYM callable services .....	486
28.9 AMODE 64 CELQPIPI service vector .....	487
28.10 AMODE 64 CEETBCK and CEEHGOTO .....	488
28.10.1 __far_jump() function .....	489
28.10.2 __le_traceback() function .....	489
28.11 XPLINK enhancements .....	489
<b>Appendix A. Metal option of XL C compiler</b> .....	491
A.1 JCL procedure METACALG .....	492
<b>Appendix B. System REXX for z/OS</b> .....	495
B.1 REXX exec WHOIAM .....	496
<b>Appendix C. z/OS Communications Server</b> .....	499
C.1 IPSEC policy configuration for SC70 .....	500
C.2 IPSEC policy configuration for SC65 .....	503
C.3 SC65 pbr configuration files .....	507
C.4 SC70 pbr configuration files .....	507
C.5 SC65 netstat -A command .....	508
C.6 SC70 netstat -A command .....	510
C.7 pasearch -R command .....	512
<b>Related publications</b> .....	515
IBM Redbooks .....	515
Other publications .....	515
Online resources .....	516
How to get Redbooks .....	517
Help from IBM .....	517
<b>Index</b> .....	519

Archived

# 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 give 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 new 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

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.

# Trademarks

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

AD/Cycle®	EPILOG®	REXX™
AIX®	FICON®	RMF™
BookManager®	GDPS®	S/390®
C/370™	HiperSockets™	System z™
CICS®	i5/OS®	System z9™
CUA®	IBM®	System Storage™
Domino®	IMS™	System/390®
DB2®	Language Environment®	SAA®
DFSMS™	Lotus®	SYSREXX™
DFSMSdfp™	MQSeries®	Tivoli®
DFSMSdss™	MVS™	VM/ESA®
DFSMSHsm™	MVS/ESA™	VTAM®
DFSMSrmm™	NetView®	WebSphere®
DFSORT™	OMEGAMON®	xSeries®
DRDA®	OS/390®	z/Architecture®
DS6000™	Parallel Sysplex®	z/OS®
DS8000™	RACF®	z/VM®
eServer™	Redbooks®	zSeries®
ELX®	Redbooks (logo)  ®	z9™

The following terms are trademarks of other companies:

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Java, Javadoc, Solaris, StorageTek, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

# Preface

This IBM® Redbooks® publication describes the functional enhancements to IBM z/OS® for Version 1 Release 9 (z/OS V1R9). These enhancements are designed to help installations install, tailor, migrate, and configure z/OS V1R9.

This book describes the new enhancements as follows:

- ▶ z/OS Version 1 Release 9 overview
- ▶ Installation and migration to z/OS V1R9
- ▶ Coupling Facility enhancements
- ▶ ICSF support for PKCS #11
- ▶ Allocation dynamic storage improvements
- ▶ System Logger enhancements
- ▶ SMF recording to log streams
- ▶ GRS enhancements
- ▶ Message Flood Automation
- ▶ Workload Manager (WLM) enhancements
- ▶ C/C ++ enhancements
- ▶ ISPF enhancements
- ▶ Security enhancements
- ▶ z/OS Communication Server
- ▶ New faces of z/OS
- ▶ System REXX™ for z/OS
- ▶ z/OS XL C Metal option
- ▶ z/OS UNIX® System Services
- ▶ SDSF enhancements
- ▶ Program management enhancements
- ▶ JES2 and JES3 enhancements
- ▶ IBM Health Checker for z/OS
- ▶ DFSMS™ enhancements
- ▶ Large format data sets
- ▶ RMF™ enhancements
- ▶ XML enhancements
- ▶ RRS enhancements
- ▶ Language Environment®

## The team that wrote this book

This IBM Redbooks publication was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Paul Rogers** is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS, z/OS UNIX, JES3, and Infoprint Server. Before joining the ITSO 19 1/2 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England for seven years, providing OS/390® and JES support for IBM EMEA. He also worked in the Washington Systems Center for three years, and has been with IBM for more than 40 years.

**Paul-Robert Hering** is an IT Specialist at the ITS Technical Support Center, Mainz, Germany who provides support to customers with z/OS and UNIX System Services-related questions and problems. He has participated in several ITSO residencies since 1988, writing about UNIX-related topics. Prior to supporting OS/390 and z/OS, Paul-Robert worked for many years with VM and all its different flavors (VM/370, VM/HPO, VM/XA, and VM/ESA®).

**Lutz Kuehner** is a z/OS IT Specialist working in IBM Global Services, Germany. He has 21 years of experience in the mainframe field and has contributed to several IBM Redbooks publications.

**Jean-Louis Lafitte** is a Senior System Architect at GATE Informatic SA, an IBM Premier Business Partner in Switzerland. He has 36 years of experience on IBM Large Enterprise Systems, has worked on different parallel machines (IBM RP3, CM2, KSR1, IBM SP1, SP2 and BG/L), and has been associated with Parallel Sysplex® since 1984. He holds a Ph.D. degree in Theoretical Computer Science and several patents in System/390® architecture. He is a member of ACM and IEEE.

**Marcos Minatto** is a System Programmer at Banco do Brasil in Brazil. He has four years of experience in the z/OS field and holds a degree in Information Systems. His areas of expertise include z/OS, installation, maintenance, capacity planning, and Workload Manager.

**Jacqueline Mourao** is a Systems Programmer at Banco do Brasil, an IBM customer. She has five years of mainframe experience and holds a degree in Information Systems. Her areas of expertise include z/OS, installation and maintenance, Parallel Sysplex and security.

**Meganen Naidoo** is a Technical Architect with Business Connexion, a large outsourcing IBM business partner in South Africa. He has more than 23 years of mainframe experience, working on VM, z/OS and Linux® system platforms, and has contributed to several IBM Redbooks publications. Meganen's areas of expertise include a variety of technical topics about z/OS, CICS® and Storage Management.

**Gil Peleg** is a z/OS System Programmer working for Tangram-Soft LTD in Israel. He has nine years of experience in mainframe systems and holds a degree in Computer Science. Gil is responsible for supporting zSeries® customers in Israel and teaching zSeries-related courses.

**Evanir Philipi** is a Certified z/OS IT Specialist working with IBM Global Services, Brazil. He has worked with IBM mainframes for 35 years. Evanir is a z/OS instructor and a z/OS Technical Leader (Banco do Brasil).

**Giancarlo Rodolfi** is a zSeries Certified Consultant TSS in Brazil. He has 21 years of experience in the zSeries field. His areas of expertise include zSeries and Linux. He has written extensively about z/OS Communication Server.

Thanks to the following people for their contributions to this project:

Rich Conway, Roy Costa  
International Technical Support Organization, Poughkeepsie Center

Ray Mansell  
Scalable Systems, IBM Research Hawthorne

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:  
[ibm.com/redbooks/residencies.html](http://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 in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto: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

Archived



# z/OS Version 1 Release 9

With z/OS V1R9, IBM extends the value of the mainframe operating system with improvements in all of its core competencies, including scalability, availability, and resource optimization. There are advancements in ease of use for both new and existing IT professionals coming to z/OS. There is an enhanced centralized policy-based tailoring for application networking and security. For Web-based applications, centralized encryption key management is introduced. There is also autonomic policy-based application performance management.

z/OS V1R9 helps to provide constraint relief and improve overall performance and/or scalability of the following items:

- ▶ Coupling Facility (CF) Duplexing
- ▶ CF performance monitoring
- ▶ GRS management of ENQs
- ▶ XCF Couple Data Set
- ▶ Language Environment heap pools

With increased focus on simplifying z/OS for IT professionals, plans for z/OS V1R9 include improvements to:

- ▶ IBM Health Checker for z/OS
- ▶ IBM Configuration Assistant for z/OS
- ▶ Communications Server
- ▶ DFSMSrmm™
- ▶ ISPF
- ▶ Hardware Configuration Manager (HCM)
- ▶ Coupling Facility services
- ▶ A new dbx GUI

# 1.1 z/OS V1R9 enhancements

As mentioned, with z/OS V1R9, IBM plans to extend the value of the flagship mainframe operating system with improvements in all of its core competencies, including scalability, availability, and resource optimization. With increased focus on simplifying z/OS for IT professionals, plans for z/OS V1R9 include improvements to the IBM Health Checker for z/OS, the IBM Configuration Assistant for z/OS Communications Server, DFSMSrmm, ISPF, Hardware Configuration Manager (HCM), and Coupling Facility services, as well as a new dbx GUI. Additional enhancements to z/OS are planned to make the operating system more powerful in applying centralized policy-based rules for defining and controlling how your applications behave.

Figure 1-1 describes the enhancements to z/OS V1R9 in terms of the type of change the new functions apply. This is more or less the roadmap for the changes to be made in z/OS in this and future releases.

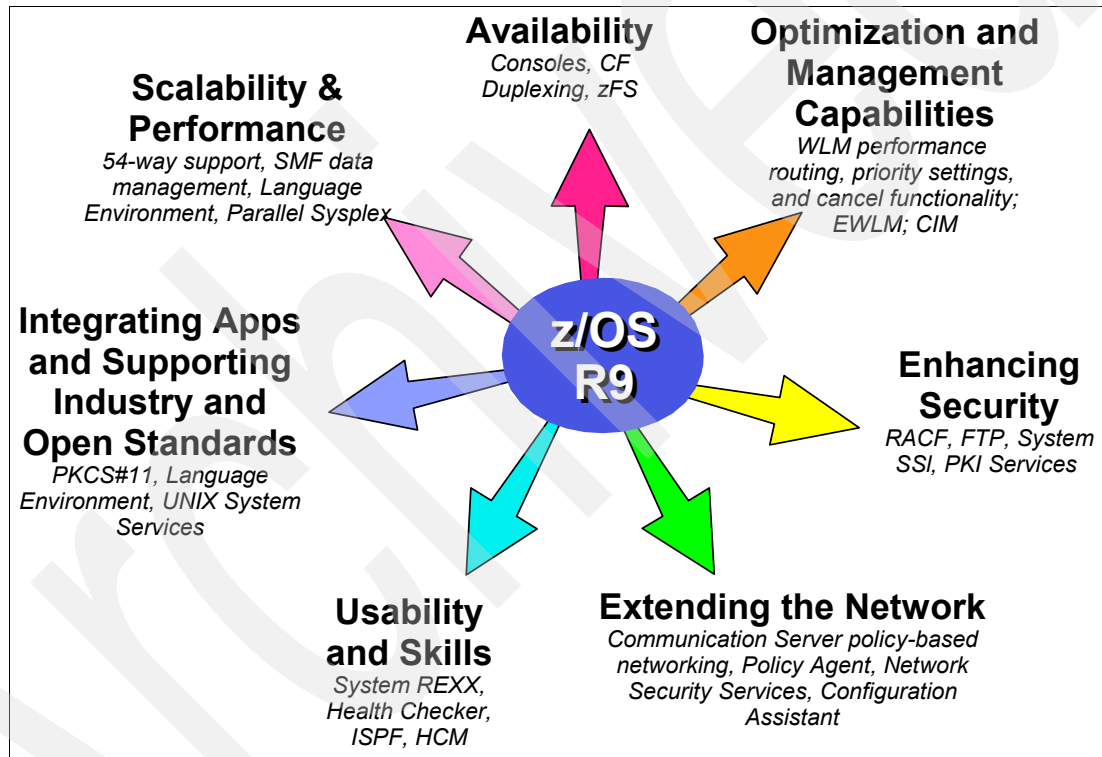


Figure 1-1 Overview of the functional areas of change with z/OS V1R9

## Scalability, performance and availability

In z/OS V1R9, enhancements are provided to extend system limits in support of system availability and data and application workload growth. Support is provided that can improve the availability of Consoles component, System Logger, z/OS UNIX System Services and z/OS UNIX File System, and Sysplex Failure Management function.

- ▶ Support up to 54-way for capacity growth and vertical scalability within a single system image. 54-way support is the sum of Central Processors (CPs), IBM System z9™ Integration Information Processors (zIIPs), and IBM System z™ Application Assist Processors (zAAPs).
- ▶ Improve consoles message handling with policy-driven Message Flood Automation.

- ▶ Allow SMF to be configured to use System Logger to write data to a log stream. This is expected to improve the scalability, collection, and the overall management of SMF data.
- ▶ Relieve virtual storage constraints for the BCP allocation, the I/O Supervisor, and DFSMSHsm™.
- ▶ Provide performance improvements for Parallel Sysplex couple data sets, CF Duplexing, z/OS UNIX File System (zFS), and Language Environment.
- ▶ In addition to the current hex and binary floating-point formats, z/OS XL C/C++ supports in z/OS V1R9 (with a possible roll-back to z/OS V1R8) the decimal floating-point formats.

Despite the widespread use of binary arithmetic, decimal computation remains essential for many applications. Not only is it required whenever numbers are presented for human inspection, but it is also often a necessity when fractions are involved. The hardware implementation of this arithmetic is expected to significantly accelerate a wide variety of applications. This support is available on the IBM System z9 BC and IBM System z9 EC, and is activated at the XL compiler in C/C++ via a new DFP option.

### **Application integration**

To embrace open and industry standards to support requirements for application portability, z/OS V1R9 has several important functions intended to extend existing applications, integrate new applications, and support industry and de facto standards.

- ▶ Adopting the PKCS#11 standard allows mainframe encryption and centralized key management to be used by Web-based applications and networking environments.
- ▶ There are improvements to LDAP enable application registries to be more easily centralized, managed, and recovered.
- ▶ There are improvements to z/OS UNIX System Services help enable porting of UNIX applications to z/OS.
- ▶ Language Environment has enhancements to language, currency, multicast source filtering, and XPLINK support.

### **Security enhancements**

z/OS V1R9 improves and extends the world-class security capabilities of the platform in the following enhancements to PKI Services, RACF®, and SAF. These enhancements improve the creation, authentication, renewal, and management of digital certificates.

- ▶ z/OS System SSL and Application Transparent-TLS are opened up to more application exploiters.
- ▶ RACF has added infrastructure for password phrase support and AES cryptography.
- ▶ z/OS Communication Server has introduced many functions for centralized security and policy-based management.

### **Optimization and management capabilities**

z/OS V1R9 offers outstanding overall resource utilization capabilities and policy-based workload management.

- ▶ The z/OS Workload Manager (WLM) is enhanced with improved performance routing, priority settings, and cancel functionality.
- ▶ z/OS supports the latest Common Information Model (CIM) standard to help z/OS to integrate with more industry tools.
- ▶ EWLM is enhanced to include Open Group's ARM 4.1 (Application Response Manager) extensions for z/OS.

## Networking

Communications Server for z/OS V1R9 enters into a new era of z/OS middleware enablement by enhancing the security and control of network communications. Security capabilities have enhancements with expanded application-transparent security for TN3270 and FTP. Control enhancements are planned in the areas of network traffic and sysplex operations. z/OS Communications Server is enhanced to include a new function: policy-based routing.

## Ease of use

With increased focus on simplifying z/OS for IT professionals, z/OS V1R9 provides improvements to the IBM Health Checker for z/OS, the Configuration Assistant for the z/OS, Communications Server, DFSMSrmm, ISPF, Hardware Configuration Manager (HCM), Coupling Facility configuration, and CF performance monitoring, and also provides a new dbx GUI. These improvements can help simplify systems management, improve system programmer and operator productivity, and make the functions easier to understand and use. System REXX (SYSREXX™) makes possible execution of REXX routines in an authorized environment. SYSREXX execs can be used to automate complex operator commands and other system functions.

## 1.2 BCP miscellaneous enhancements

The maximum specifiable size of the MVS™ system trace is changed from the current value of 999 K per CPU. The practical maximum will be in the order of many MBs per CPU. It will vary depending on the size of the LPAR and the applications that contend for real storage.

## 1.3 z/OS support for IBM System z servers

Starting with z/OS V1R6, up to 32 processors are supported in a single logical partition on IBM System z9 EC and z990 servers. With z/OS V1R9, support is provided for z/OS to run up to 54 processors in a single logical partition on z9 EC servers.

**Note:** The total number of processors defined in a z/OS logical partition is the sum of general purpose processors (CPs), System z9 Application Assist Processors (zAAPs), and System z9 Integrated Information Processors (zIIPs).

## 1.4 z/OS UNIX System Services

In z/OS V1R9, enhancements for z/OS UNIX System Services improve management of automount file systems that are managing a directory located in an automove (unmount) file system. The automount file system now inherits the automove (unmount) attribute rather than mounting as automove (yes). Note that IBM Health Checker for z/OS flags this inconsistency when automount is mounted as automove (yes).

### SUSv3 standard

To enable and protect customer investment by addressing open and de facto standards, UNIX03, also known as Single UNIX Specification Version 3 (SUSV3), is the latest UNIX standard ratified by the Open Group and then by other standards organizations in 2001 and 2002. This newest standard defines C/C++ APIs needed by Enterprise Applications workloads. Although z/OS UNIX has not pursued compliance with a new UNIX standard since

OS/390 Release 2, new incremental functions are provided as required for applications such as SAP® and PeopleSoft®. The functions mandated by the SUSV3 standard will allow z/OS to host customer applications and allow ISVs greater portability and flexibility.

- ▶ These utility enhancements provide function available on other UNIX platforms.
- ▶ The UNIX system services kernel effort is in support of the Language Environment initiative.
- ▶ Completion of the pthread functions will benefit applications porting to z/OS.
- ▶ Changes in the Language Environment run-time library allow applications to target SUSv3 and XSI environments, and support SUSv3 and XSI behaviors during run-time where differences exist.

## **z/OS UNIX file system**

Enhancements are also provided for the z/OS UNIX file system to make the following reliability, availability, and serviceability improvements.

- ▶ Record UNIX file and directory deletion with a new subtype of the SMF type 92 records.
- ▶ In a shared file system configuration, provide more consistent (and predictable) file system shutdown/recovery behavior based on the file system AUTOMOVE setting. In prior releases, the AUTOMOVE specification is not honored if the file system is mounted in a mode in which the physical file system (PFS) provides “sysplex-aware” capability.
- ▶ The F BPXOINIT,FILESYS=FIX command adds support to allow the FIX option to detect and correct CDS serialization state information when MEMBER GONE (failed system recovery) processing is in progress. Do not attempt to perform file system-specific recovery.
- ▶ The F BPXOINIT, RECOVER=LATCHES command now takes multi-address space multi-system dumps for file system problems. Logic is added in file system mainline paths to detect when PFS operations are outstanding, and provide multi-address space and the multi-system dump utility that handles RECOVER=LATCHES to capture file system SVC dumps. In doing this, it reduces use of the mount latch and enhances RECOVER=LATCHES to terminate system tasks in some circumstances.
- ▶ The USS file system provides for I/O completion notifications via standard message queues. This gives an unauthorized application complete control over what thread in a process receives notification of Asyncio completions.
- ▶ Enhancements to some z/OS UNIX commands

Enhancements to some z/OS UNIX commands are intended to help enable the porting of UNIX applications and shell scripts to the z/OS platform and the development of portable applications. The enhancements include changes to the following commands:

awk, bc, ed, file, mailx, od, sed, tr, uuencode, and uudecode

## **SMF record 92**

Within the UNIX System Services file system, SMF record 92 is enhanced for record-to-record file deletes to simplify system administration tasks. This is in response to multiple customer requests for this support. The SMF component of the BCP element that collects data for accounting is enhanced to process SMF job and job-step accounting records by identifying processes by specific identifiers. The SMF file system records currently describe numerous file system events (such as file open, file memory map, file system mount, change of ownership). Therefore, a subtype 14 record is being added to the SMF 92 records for reporting file and directory deletion.

## USS kernel

USS Kernel provides support for `pthread_cancel`. It allows `pthread_cancel()` to terminate non-USS lineage threads or suspended threads with FRRs.

## zFS file systems

zFS shared file system performance is improved by its becoming sysplex-aware for read-write file systems. zFS and HFS are already sysplex-aware for file systems that are mounted read-only. Because zFS is sysplex-aware, USS now sends file requests directly to the zFS physical file system (PFS) without forwarding them to the USS file system owner. zFS uses XCF communications to forward requests to the owning zFS PFS when necessary. In addition, end-of-memory (EOM) recovery support for the zFS sysplex code enables zFS to recover from end-of-memory conditions that occur while executing the new zFS code added for zFS sysplex file.

zFS support has been implemented to handle NFS V4 share reservations for when zFS runs in sysplex-aware mode using its token management mechanism. This removes restrictions for the z/OS NFS V4 server when running zFS file systems in a shared file system environment. It also improves performance for the z/OS NFS V4 server when using zFS file systems in a shared file system environment.

## 1.5 z/OS Workload Manager

The z/OS Workload Manager (WLM) is enhanced with improved performance routing, priority settings, and cancel functionality, further improving on the mainframe's leadership position in workload management capabilities. With z/OS WLM, you can define business and performance goals customized for your applications. The z/OS system decides how much resource, such as CPU and storage, should be given to applications that serve the workload to meet the goal.

WLM constantly monitors the system and adapts resource applications to meet application goals. It takes into account not only server resources but also network traffic, router bottlenecks, application health, and transaction prioritization as well, thus providing autonomic, policy-based z/OS performance management that can be tuned to meet the needs of your applications.

WLM is changed to increase the priority of canceled address spaces. This is expected to help them be terminated more quickly. This can eliminate the need to reset the priority of a canceled job, task, or user to speed address space termination when resolving resource contention issues.

WLM adds a new parameter on the IWMSLIM service, which allows the server region to tell WLM that a number of minimum server regions should be started in parallel. The new parameter can allow applications to control whether WLM should start server regions in parallel or sequentially.

WLM is enhanced for discretionary work. During periods of 100% CPU utilization, it is possible that discretionary workloads (workloads defined by your installation to have lower dispatch priority) will not be dispatched for execution. These discretionary workloads may obtain and hold serially reusable resources required by other workloads, which may block the progress of higher dispatch priority workloads.

In z/OS V1R9, it is possible to specify that any address spaces and enclaves that have work that is ready to run but do not get CPU service within a certain time interval can be

temporarily promoted to a higher dispatch priority. RMF supports this function by reporting relevant measurements.

The EWLM application response measurement (ARM) V4.1 support implements extensions to provide z/OS support for monitoring applications based on an asynchronous messaging model. ARM V4.1 is currently a draft standard and is expected to be published by the Open Group. Additional extensions for asynchronous messaging are provided for applications running under CICS using the WLM delay monitoring services.

The WLM routing services are enhanced to recognize the zAAP and the zIIP capacity of a System z server.

## 1.6 Console message flood

In z/OS V1R9, the consoles component is integrating the message flood automation function that was made available via APAR OA17514 for z/OS V1R6 and higher. Message flood automation provides a specialized, policy-driven automation for dealing with high volumes of messages occurring at very high message rates. The policy can be set in a PARMLIB member and examined and modified through operator commands.

## 1.7 System Logger

System Logger with z/OS V1R9 improves availability by providing support for log stream data set asynchronous recalls that allows for multiple, concurrent, migrated data set recall requests to be processed by System Logger.

## 1.8 SMF use of System Logger

z/OS V1R9 enhances SMF data management by allowing SMF to be configured to use System Logger to write data to a log stream. This is expected to allow the system to support far higher data write rates than can be supported when using SYS1.MAN data sets when the Coupling Facility (CF) is used. The use of DASDONLY log streams is also supported.

Also, this design allows a specification that different SMF record types be written to separate log streams, for which different retention periods can be specified. This can help improve both scalability and SMF data management.

## 1.9 Coupling Facility enhancements

z/OS V1R9 includes support for a number of usability enhancements to the CF structure, as described here.

### **REALLOCATE process**

The **SETXCF START, REALLOCATE** system command was enhanced originally for z/OS V1R4 and higher via APAR OA08688. The REALLOCATE process itself provides a simple, easy-to-use mechanism for dynamically optimizing the placement of CF structures among the CFs in a Parallel Sysplex. It determines the “most preferred” CF locations for the CF structure

instances based on the CFRM policy and current conditions, and serially moves the structures to those most preferred CFs in a nondisruptive fashion.

With z/OS V1R9, REALLOCATE now includes the following enhancements:

- ▶ A structure-level CFRM policy control allows selected structures to be bypassed by REALLOCATE processing, if necessary
- ▶ Support is included to automatically initiate duplexing for CF structures that should be duplexed
- ▶ The capability is included to complete a pending policy change for structures without rebuilding the structure, whenever possible
- ▶ Also included is improved processing of structures which make use of the exclusion list (EXCLLIST) option in the CFRM policy

### CF maintenance mode

z/OS V1R9 includes support for placing Coupling Facilities into a new state known as *maintenance mode*. When a CF is in maintenance mode, it is logically ineligible for CF structure allocation purposes, as if it had been removed from the CFRM policy entirely (although no CFRM policy updates are required to accomplish this). Subsequent rebuild or REALLOCATE processing will also tend to remove any CF structure instances that were already allocated in that CF at the time it was placed into maintenance mode.

In conjunction with the REALLOCATE command, the new maintenance mode support can greatly simplify operational procedures related to taking a CF down for maintenance or upgrade in a Parallel Sysplex. In particular, you no longer have to laboriously update or maintain several alternate copies of the CFRM policy that omit a particular CF which is to be removed for maintenance.

**Note:** RMF is planned to provide information about the CF processor resources consumed by each Coupling Facility (CF) structure. This information is provided by both Postprocessor and Monitor III. These enhancements are intended to allow better CF performance monitoring and problem determination by tracking utilization at a CF structure level. A CF level 15 is required, which is supported on System z9 servers.

### Synchronization protocols

The Coupling Facility-to-Coupling Facility (CF) synchronization protocols for CF Duplexing have been streamlined, thus resulting in improved performance (service time) and throughput for duplexed requests that can take advantage of this enhancement. This enhancement can help reduce the overhead of CF Duplexing, and may help make duplexing a more viable alternative for use in providing high availability for CF structure data. A new CF level is required, which is planned to be supported on System z9 servers.

## 1.10 GRS 64-bit exploitation

GRS is enhanced to further exploit 64-bit addressing. This can dramatically increase the number of concurrent enqueues that can be supported on a z/OS system. This new function extends the GRS existing 64-bit support for Star-mode control blocks. With z/OS V1R9, the majority of GRS enqueue-related control blocks reside in storage above the 2 GB bar. This support is supported for all three GRS modes, None, Ring, and Star. This should significantly reduce CMSEQDQ lock hold time and provide much better performance for ISGENQ and ENQ/DEQ/RESERVE LINKAGE=SYSTEM users.



In addition, GRS is enhanced to insure that future demands for more concurrent ENQs are successfully satisfied.

## 1.11 Sysplex failure management

Sysplex failure management (SFM) allows you to define a sysplex-wide policy that specifies the actions that MVS is to take when certain failures occur in the sysplex. A number of situations might occur during the operation of a sysplex when one or more systems need to be removed so that the remaining sysplex members can continue to do work. The goal of SFM is to allow these reconfiguration decisions to be made and carried out with little or no operator involvement.

The sysplex failure management function in z/OS V1R9 is enhanced to support a new policy specification for how long a system should be allowed to remain in the sysplex when it appears unresponsive because it is not updating its system status on the sysplex couple data set, yet is still sending XCF signals to other systems in the sysplex. A system that is in this state is definitely not completely inoperable (since it is sending XCF signals), and yet it may not be fully functional either, so it may be causing sysplex sympathy sickness problems for other active systems in the sysplex.

The new SFM policy externally provides a way for installations to limit their exposure to problems caused by such systems, by automatically removing them from the sysplex after a specified period of time.

This new SFM function supports a new policy specification to indicate that, after a specified period of time, the system may automatically terminate XCF members which have been identified as stalled and who also appear to be causing sympathy sickness problems. If allowed to persist, these stalled members can lead to sysplex-wide hangs or other problems, not only within their own XCF group, but also for any other system or application functions that depend on the impacted function. Automatically terminating these members is intended to provide improved application availability within the sysplex.

## 1.12 Program management binder

The binder has been enhanced as follows:

- ▶ A C front-end to binder APIs is designed to simplify using both the regular binder APIs and fastdata APIs for C and C++ programmers.
- ▶ Definition Side-Files in z/OS UNIX archives are intended to allow programmers to package their Dynamic Link Library (DLL) side-decks in UNIX archive files produced by the ar utility.
- ▶ The fastdata API rewrite is designed to provide improved reliability for fastdata APIs.
- ▶ Improvements to AMBLIST XREF are intended to provide improved execution time and capability when processing cross-reference information of large programs.
- ▶ RECFM=U verification is designed to provide the same protection against writing programs into non-program PDS libraries as is provided for PDSE libraries.
- ▶ A new binder INFO option will list all installed PTFs in the binder SYSPRINT output.
- ▶ The binder is modified to recognize and process definition side-decks (which contain IMPORT control statements), which are members of z/OS UNIX archive files.

- ▶ The binder is modified to optionally provide a module map as part of a program object or load module, for the use by dbx, potentially other debuggers, or module map type utilities. Equivalent information can be obtained via multiple binder API calls, but keeping the information in the loaded part of the module, and in one place, is faster and is usable when the module itself is not available (as in IPCS dumps). It is an optional feature, because it increases the virtual storage size of the loaded module (this is expected to be particularly true of C++ programs).
- ▶ The binder is modified to enhance the CHANGE and REPLACE control statements with a new IMMED option which will cause them to operate against any content previously included in the module being bound. This allows more freedom in the positioning of those control statements and enables them to be more easily used to alter the contents of a multi-object input file. This was previously available only when using the API.

## 1.13 XCF Couple Data Set

z/OS V1R9 includes support for improved parallelism in XCF Couple Data Set access channel programs for all supported types of Couple Data Sets. By more granularly expressing whether or not a particular channel program intends to update any data, channel programs that could not have run in parallel previously will now be able to do so, resulting in improved I/O performance and throughput. This enhancement was originally made available for z/OS V1R4 and higher with APAR OA15409.

In z/OS V1R9, enhancements for the `D XCF, COUPLE, TYPE=BPXMCD`s command include the current defined values for MAXSYSTEMS, MOUNTS and AMTRULES for the TYPE(BPXMCD) couple data set. Because these values can be updated dynamically, it will be easier to keep track of changes that could impact the shared file system configuration.

## 1.14 Language Environment

Language Environment enhancements help improve the performance of applications using heap pools in the following manner:

- ▶ Allow the heap pool control data and individual cells to be aligned, to better optimize use of the processor cache
- ▶ Eliminate stack transitions in an XPLINK environment
- ▶ Design heap pools cells to always align cells on a 16-byte boundary under AMODE 64

Language Environment now also eliminates stack transitions during long division and long multiplication in an XPLINK environment.

The Language Environment CEEBLDTX utility is now designed to run in a UNIX shell command. This utility is made available as a shell command.

The CICS CLER function is enhanced for displaying and modifying Language Environment run-time options.

Implementing the functionality of CEE5DLY on z/OS now enables LE-conforming programs to suspend execution and then allow the code using ILBOWAT to be replaced without requiring in-house assembler code.

Language Environment now provides AMODE 64 equivalents of the 31-bit callable service, CEETBCK, which is a callable service that assists in tracing the call chain backwards, and of

CEEGOTO, which is a callable service that provides the capability to perform a “goto” to a known location within the program stack.

In addition, Language Environment also provides support for XPLINK applications running as IMS™ transactions. Support is added for IMS regions running with or without Language Environment Library Routine Retention (LRR) active.

In z/OS V1R9, CELQPIPI (the Language Environment pre-initialization facility for AMODE64) provides the service exit points currently available in 31-bit CEEPIPI (minimally, the program LOAD/UNLOAD service), in AMODE64.

New locales available from CLDR Repository are added to make z/OS more usable with as many languages as possible. Additionally, ASCII versions for a set of existing euro and pre-euro locales will be added. Locales for Turkey are updated to make use of the new Turkish Lira currency.

## 1.15 DFSMS enhancements

DFSMS provided support for DSNTYPE=LARGE data sets which can contain more than 65,535 tracks beginning with z/OS V1R7. In z/OS V1R9, the following TSO/E functions are updated in order to utilize this support:

- ▶ LISTDSI, used in REXX execs and CLISTs
- ▶ PRINTDS command
- ▶ TRANSMIT and RECEIVE commands

The serialization used by DFSMSrmm for its CDS is changed to use a new resource name that includes the CDS ID. This avoids conflicts when multiple RMMplexes run in the same sysplex.

### DFSMSdfp

DFSMSdfp™ OAM (Object Access Method) introduces two new sublevels into the tape level of the OAM storage hierarchy. This effectively expands OAM's storage hierarchy into four levels: disk, optical, tape sublevel 1 (TSL1), and tape sublevel 2 (TSL2). In addition to enabling the ability to write and read object data directly to and from a given sublevel, this support provides the ability to transition object data within the tape family (for example: from VTS to native tape) during an OSMC storage group cycle. Prior to this support, data movement within the tape family could only be accomplished manually via the **MOVEVOL** or **RECYCLE** commands.

z/OS V1R9 has enhancements to Access Method Services which allow data set name masking on delete requests.

### DFSMShsm

The amount of storage that DFSMShsm uses below 16 MB is now reduced in an effort to address storage-related abends (878, 80A, and so on). Additional fields are added to the DFSMShsm function statistics records (FSRs) to improve the data available for statistical analysis of the DFSMShsm environment.

### DFSMSrmm

DFSMSrmm interaction with system managed volumes in an IBM system managed library is improved through multiple changes which are expected, especially in larger VTS installations, to result in shorter elapsed time and more flexibility during inventory management.

DFSMSrmm is enhanced so that you can now control long-running local subsystem requests. These requests can be ended, held, and released. This enhancement will enable better management when required either by system automation or by the operator because of operational priorities.

The DFSMSrmm CIM provider code is updated to support OpenPegasus CIM Server with 2.5.1 and the subclasses supported are planned to be extended to cover all DFSMSrmm-managed resources.

System symbols are now supported for DFSMSrmm parmlib members. This enhancement is designed to allow sharing of DFSMSrmm parmlib members more easily. Additionally, indirection can be used to point to another parmlib member that might contain system-specific options.

DFSMSrmm SEARCH subcommands with CLIST are enhanced so that it is possible to optionally append to an existing CLIST data set. Almost any format of CLIST data set is supported, and the subcommands support a way to break the results into chunks for easier results management.

DFSMSrmm is enhanced to support almost any unqualified data set name up to 44 characters. The product version can be alphanumeric and volumes and data sets can be declassified.

### **Network File System V4 client**

z/OS V1R9 now supports the Network File System (NFS) V4 industry-standard protocol. This maintains z/OS NFS client competitiveness with the clients provided by other platforms by allowing the exploitation of the NFS v4 protocol.

The NFS V4 client support in previous versions accessed shared file systems by mounting them from an NFS server machine. Mount was used to obtain the filehandle. Subsequent locking was done by the network lock manager (NLM).

With this release, NFS Version 4 integrates these disparate elements into a single protocol that uses a well-defined port that enables NFS to more easily transit firewalls to enable support for the internet. The single, integrated protocol eliminates the need for the ancillary mount and NLM protocols, and introduces the concept of mandatory locking in addition to advisory locking and a host of new verbs (like OPEN and CLOSE) which can be nested within the RPC COMPOUND to reduce network round-trip latency.

NFS RAS and constraint relief improvements in V1R9 include the following enhancements:

- ▶ Establish z/OS NFS EAL4 compliance by creating low level design documents for all security-related code paths to aid auditors in evaluating z/OS NFS compliance with the EAL4 standard, and implement data labeling support in the z/OS NFS server (NFSS).
- ▶ Implement 24-bit addressing relief in the z/OS NFS server, eliminating 24-bit addressing restrictions wherever possible. This implementation is available for z/OS NFS V1R7 and V1R8.
- ▶ NFS serviceability improvements include updating the z/OS NFS server Ctrace. This update of the Ctrace function is done to exploit the enhanced capabilities developed as part of the z/OS NFS Client Ctrace deployment in z/OS V1R8 NFS.

The separate GFSCMAIN NFS client load module is eliminated by merging it with the GFSCINIT load module.

An **AddDS** operator command implements a new set of **F NFS, xxxxx** operator commands to support the dynamic replacement of the NFS server mounthandle database (MHDB) and locking database (LDB) data sets without requiring an NFS server restart.

- ▶ The z/OS NFS V4 RPCSEC\_GSS security function is enhanced to support multi TCP/IP stack system configurations, eliminating the single stack restriction.

## 1.16 z/OS Communications Server

Additional enhancements to z/OS make the operating system more powerful in applying centralized policy-based rules for defining and controlling how your applications behave. They also include the planned ability to provide centralized policy services and policy-based routing (along with the ability to apply security, intrusion detection/defense, qualities of service, as well as other features). z/OS can help customize the network to suit the needs of applications, and it does so in a simplified, centralized, manageable, and auditable manner that is anticipated to be transparent to the application.

With z/OS V1R9, CSA/ECSA usage has been reduced for VTAM® by taking the following actions:

- ▶ Reducing the size of the ACDEB used by Telnet
- ▶ Moving the ISTRIVL that is provided to the ACB Monitor exit to VTAM private
- ▶ Eliminating the usage of the FMCB/FMCB extension for most SSCP-LU sessions

Communications Server for z/OS V1R9 is providing new capabilities for z/OS middleware enablement by enhancing the security and control of network communications. Security is changed by providing expanded application-transparent security for TN3270 and FTP. Control enhancements are made in the areas of network traffic and sysplex operations.

### Policy-based routing

z/OS Communications Server is enhanced to include a new function: policy-based routing. Policy-based routing enables the TCP/IP stack to make routing decisions that take into account criteria other than simply the destination IP address/subnet (as is done with both static and dynamic routing). The additional criteria can include the job name, source port, destination port, protocol type (TCP or UDP), source IP address, NetAccess security zone, and security label.

With policy-based routing, you can define policy to select the network that will be used for outbound traffic based on the application originating the traffic. The IBM Configuration Assistant for z/OS Communications Server is enhanced to support policy-based routing.

With policy-based routing, you can define policy to select the network that will be used for outbound traffic based on the application originating the traffic. To provide dynamic routing characteristics (for example, best route calculation, route availability awareness, and so on) to policy-based routing, the policy-based routing is built on top of a dynamic routing foundation. The policy-based routing function interfaces with OMPROUTE to obtain dynamic routes to populate the policy-based route tables.

**Note:** Policy-based routing applies only to IPv4 TCP and UDP traffic that originates at the TCP/IP stack. Traffic using protocols other than TCP and UDP, all traffic being forwarded by the TCP/IP stack, and all IPv6 traffic, will always be routed using the main route table, even when policy-based routing is in use.

### Policy Agent

Policy Agent is enhanced to take on additional roles that support the goal of centralized policy management. Policy Agent can be configured to act as a policy server. In this role, it not only

reads and installs local policies for a set of TCP/IP stacks, but also loads policies on demand for policy clients. This allows all policies for a set of systems to be administered on a single system.

Policy Agent can be configured to act as a policy client. In this role it connects to the policy server and retrieves remote policies that are then installed in the local TCP/IP stacks. The choice of local or remote policies may be made for each policy type (AT-TLS, IDS, IPSec, QoS) and for each TCP/IP stack.

**Note:** z/OS V1R9 Communications Server no longer supports the LDAP protocol Version 2 for Policy Agent communication with an LDAP server.

For QoS and IDS LDAP policies in Policy Agent, use an LDAP protocol Version 3 server. LDAP protocol Version 3 has improvements in internationalization, authentication, referral, and deployment.

### **IBM Configuration Assistant for z/OS Communications Server**

The IBM Configuration Assistant for z/OS Communications Server is extended to include support for policy-based routing (PBR) and Network Security Services (NSS) configuration. This support allows an administrator to configure IPSec, Application Transparent TLS, QoS, IDS, and PBR policy using a consistent user interface.

Other new function in the IBM Configuration Assistant for z/OS Communications Server allows the configuration information for all of these technologies to be managed collectively, providing health check operations designed to insure consistent configuration across the supported technologies. The configuration information can be saved and accessed on a z/OS system or on a Windows®-based file system. The Configuration Assistant for z/OS Communications Server is a separate download.

### **Multicast support**

z/OS Communications Server has enhanced its multicast support to allow an application to filter the datagrams it receives based on the source address. z/OS Communications Server also supports the following:

- ▶ Support new APIs to allow applications to specify source filter lists. This allows the local system to filter on source addresses even if the system is not attached to a multicast router, which supports source address filtering.
- ▶ Host support for IGMPv3 and MLDv2. The system responds to queries from multicast routers and reports the source filter state of each interface.

Multicast datagrams destined for the specified multicast address coming over the specified interface are given to the application. However, all multicast datagrams which meet that criteria, regardless of the source address, are delivered to the application. This architecture is now referred to as any-source multicast (ASM).

It is possible for multiple multicast servers to be sending out datagrams for the same multicast address. An application can receive multicast datagrams from servers which it was not intending to receive. To avoid such collisions, an extension to the original ASM model was developed called source-filtered multicast (SFM). SFM specifies a set of API extensions which allow an application to filter the datagrams it receives based on the source address.

To complement these new source filtering APIs, new versions of the Internet Group Management Protocol for IPv4 (IGMPv3) and the multicast listening discovery protocol for IPv6 (MLDv2) are supported. With these new versions of the protocols, multicast routers are informed of the source IP filtering of any applications on a system. This allows the multicast

router to send only multicast datagrams that the system has applications interested in receiving. Even if a system is not attached to a multicast router which supports IGMPv3 or MLDv2, an application can still filter by source address. However, the local system will have to discard any unwanted multicast datagrams.

**Note:** z/OS Communications Server does not support any multicast routing protocols and therefore does not support any multicast routing functions of IGMPv3 or MLDv2.

## FTP and Unicode pages

FTP now supports for more Unicode code pages for file storage and file transfer. For file transfer, FTP is planned to add support for code pages UTF-16, UTF-16LE, and UTF-16BE. For file storage, FTP supports code page UTF-16. FTP always stores Unicode files in big endian format.

## TN3270E Telnet server

Prior to z/OS V1R6, the TN3270E Telnet server runs as a subtask of the TCPIP address space. In z/OS V1R6 through z/OS V1R8, users can run the TN3270E Telnet server as a separately started address space from TCPIP, or continue to run the TN3270E Telnet server as a subtask of the TCPIP address space. In z/OS V1R9, the TN3270E Telnet server is supported only when run in its own address space.

## Enterprise Extender

Enterprise Extender (EE) autonomics provides two potential performance enhancements in z/OS V1R9:

- ▶ It enables VTAM to learn of changing MTU sizes associated with an Enterprise Extender connection. With this knowledge, VTAM can avoid packet fragmentation when the MTU size is decreased. Also, in some instances, this function allows VTAM to pass larger packets to TCP/IP to better utilize the current interface associated with an EE connection.
- ▶ It provides more optimal routes for existing Enterprise Extender connections when new routes are learned by TCP/IP.

The Enterprise Extender (EE) logical data link control (LDLC) inactivity timer function is controlled by three LDLC timer operands coded on PORT definition statement. These LDLC timer operands (LIVTIME, SRQTIME and SRQRETRY settings) apply to all EE connections. In z/OS V1R9 Communications Server, the LDLC inactivity timer function has been enhanced to support unique inactivity timer settings for each local IP address (static VIPA). As a result, the LDLC timer operands may now be specified on either the PORT or on individual GROUP definition statements.

Summary information about Enterprise Extender connections is reported on the network management interface (NMI). The EE summary global record contains LDLC timer information as specified or defaulted on the PORT definition statement. In z/OS V1R9 Communications Server, the EE summary IP address record has been enhanced to contain LDLC timer information specified on the GROUP definition statement.

## High performance routing

The high performance routing (HPR) function is enhanced in z/OS Communications Server V1R9 in three ways, as follows:

1. Various inefficiencies within the HPR path switch logic are enhanced to reduce unnecessary processing and optimize code paths.
2. Redundant HPR path switch messages are reduced. The HPR path switch message reduction function provides performance and diagnostic improvements for large

installations which have hundreds or thousands of RTP endpoints on z/OS Communications Server. Performance gains are achieved by saving CPU cycles through the reduction of the number of HPR operator console messages that occur for large scale path switch events. Diagnostic procedures are improved by providing an organized, easy-to-read summary of the path switch events which took place, allowing you to more easily determine the scope and size of an outage.

This function is not associated with the message-flooding prevention table. For predictable results, the IST1494I message should not be coded in the message-flooding prevention table when the HPR path switch message reduction function is enabled.

3. Improvements to HPR activation and deactivation messages are provided. HPR activation and deactivation messages have been enhanced, by providing you additional information which should be useful when diagnosing RTP problems.

When an RTP pipe is activating, you will now receive a message group identifying the RTP PU name, whether this is the active or passive end of the pipe, the partner CPNAME, priority, associated APPNCOS and the APPN route the pipe is traversing. When an RTP pipe is deactivating, you will now receive a message group identifying the RTP PU name, whether this is the active or passive end of the pipe, and the partner CPNAME, along with the priority and associated APPNCOS.

### **TSO VTAM support**

Visibility to the CGCSGID for a TSO session is provided. Users may want to use this information as criteria to permit or deny access to an application through use of a logon exit. TSO/VTAM supports a GTTERM macro that the user can use to acquire information about a terminal. A new keyword, CODEPG, has been added to the GTTERM processing to allow the user to retrieve the CGCSGID in use for a TSO session. The SNA TSOUSER display has been enhanced to report the CGCSGID in use for a TSO user.

### **Generic resource resolution preferences**

Generic resource resolution preferences are used to control the distribution of sessions to generic resources. These preferences previously could only be set globally in the generic resource exit. This function allows you to use VTAM definitions to customize generic resource resolution preferences for individual generic resources. In addition, a new generic resource resolution preference is also being introduced that allows a generic resource resolution during third-party initiated (CLSDST-PASS) sessions to favor a generic resource on the origin host of a session.

### **MPC groups**

In z/OS V1R9 CS, activations of MPC groups that fail to meet the one read/one write requirement are put on hold, provided any needed read and/or write subchannel is an offline CTC or one that has no valid path available to the connecting host. The hold continues until all the needed subchannels come online or the group is deactivated.

New messages signal when the hold begins and when activation resumes. The display of an MPC group indicates when its activation is on hold.

Other existing output in that display identifies the offline subchannels, so appropriate action can be taken to bring enough of them online to cause activation of the MPC group to complete. FICON® CTC recovery is enabled by default, with a new start option (MPCACT) that can be used to disable the function whenever manual retry is desired.



## 1.17 z/OS security

z/OS V1R9 has security improvements on the z/OS platform in the following areas:

- ▶ PKI Services
- ▶ RACF for added infrastructure for password phrase support and AES cryptography
- ▶ SAF to help improve the creation, authentication, renewal, and management of digital certificates
- ▶ z/OS System SSL
- ▶ Application Transparent-TLS are opened up to more application exploiters
- ▶ z/OS Communications Server for centralized security and policy-based management.

### ICSF and PKCS #11 standard

The z/OS Integrated Cryptographic Service Facility (ICSF) is enhanced to include the PKCS #11 standard. ICSF is the fundamental base of z/OS mainframe encryption which enables you to encrypt and decrypt data, generate and manage cryptographic keys, and perform other cryptographic functions dealing with data integrity and digital signatures. With adoption of the PKCS #11 standard, the strength of mainframe encryption and secure centralized key management can be brought to and used by Web-based application and networking environments more easily.

This ICSF support of PKCS #11 provides an alternative to the IBM Common Cryptographic Architecture (CCA), and broadens the scope of cryptographic applications that can make use of zSeries cryptography. RACF provides PKCS#11 support with the RACF **RACDCERT** command to provide token management of certificate, public key, and private key objects.

Public Key Cryptography Standards (PKCS) is offered by RSA Laboratories of RSA Security Inc. PKCS #11, also known as Cryptoki, is the cryptographic token interface standard. It specifies an application programming interface (API) to devices, referred to as *tokens*. The PKCS #11 API is an industry-accepted standard commonly used by cryptographic applications. PKCS #11 applications developed for other platforms can be recompiled and run on z/OS.

### RACF password phrase

In z/OS V1R9, an extension is added to the password phrase that was introduced in z/OS V1R8. The minimum length of a password phrase has been lowered from 14 characters to 9. Password phrases from 9 to 13 characters in length can be used in conjunction with a new password phrase exit (ICHPWX11) which you can write to determine whether to accept them. A sample exit is provided, which uses the new System REXX facility to call a REXX exec in which you can code password phrase quality rules. A sample REXX exec is also provided. Also, password change logging and enveloping functions are extended to include RACF password phrases.

### AES cryptographic algorithm

The z/OS Network Authentication Service is enhanced to support the AES cryptographic algorithm. This support enhances interoperability with other Kerberos implementations by extending the z/OS cipher suite. Because RACF can act as the registry for the z/OS Network Authentication Service, RACF provides the management interfaces for cryptographic keys. RACF commands are planned to be extended to allow the specification of AES as a supported cipher.

## **LDAP directory server**

z/OS V1R9 enhances the LDAP directory server, known as IBM Tivoli® Directory Server for z/OS. This new server enables an installation to collapse user registries typically used by distributed applications on z/OS. This can help to simplify enterprise management and disaster recovery.

The existing Integrated Security Services LDAP Server continues to be available in z/OS V1R8 in addition to the new IBM Tivoli Directory Server for z/OS. You must apply the enabling PTF for APAR OA19286, when available, and all of its prerequisite APARs to use this function.

## **z/OS Communications Server**

z/OS Communications Server provides a new Network Security Services function to centralize certificate services, monitoring and management for IPsec security across z/OS systems within and across sysplexes. Network Security Services allows IPsec certificates to be kept in a single location, rather than having them reside on each z/OS node. The z/OS Communications Server IKE daemon is planned to be enhanced so that it can be configured to act as a Network Security client. Configuration is on a per-stack basis, such that each NSS-enabled stack will appear to the Network Security Server as an independent client. For TCP/IP stacks that are not configured to use Network Security Services, the IKE daemon will continue to manage certificates out of a local key ring.

The FTP server, FTP client, and TN3270 server now use Application Transparent TLS (AT-TLS) to manage TLS security. AT-TLS supports several security functions that the FTP server, FTP client, and TN3270 servers do not. In addition, AT-TLS provides improvements over TLS implemented by the FTP server and client which are intended to improve performance. Security defined in the TN3270 server profile and FTP.DATA continues to be available.

## **1.18 Spool Display**

Spool Display (SDSF) is being enhanced to add the capability to provide access to SDSF functions through REXX variables. The variables will be loaded with data from the SDSF panels, enabling scripts to access the data programmatically. The data can also be changed; this provides a capability similar to action characters and overtyping.

## **1.19 System REXX**

System REXX (SYSREXX) is a component that makes possible the execution of REXX routines in an authorized environment. SYSREXX execs can be used to automate complex operator commands and other system functions. SYSREXX execs can be invoked by a program interface, and by operator command. This new component is made available for z/OS V1R8 via a Web deliverable.

## **1.20 IBM Health Checker for z/OS**

IBM Health Checker for z/OS now supports checks that are written in REXX using the SYSREXX function. Also, health checks are planned for z/OS UNIX, TSO/E, the virtual storage manager component of the z/OS BCP, and z/OS Communications Server.

## 1.21 Alternate Library for REXX

New in z/OS V1R9 is the Alternate Library for REXX that enables users who do not have the REXX on zSeries library installed to run compiled REXX programs. It contains a language processor that transforms the compiled programs and runs them with the REXX interpreter which is shipped as part of the z/OS operating system.

With this implementation, software developers are no longer required to distribute the Alternate Library for REXX with their compiled REXX programs. Installations that have the REXX on zSeries Library installed may see the performance benefits of running compiled REXX. Installations that use the Alternate Library for REXX may still run the programs as interpreted.

### Benefits with Alternate Library for REXX

By including the Alternate Library for REXX with z/OS, software developers gain the benefits of shipping compiled REXX programs without the source code, as follows:

- ▶ Maintenance of the program is simplified, because the code cannot be modified inadvertently.
- ▶ Compiled programs can be shipped in load module format, thus simplifying packaging and installation.
- ▶ The Alternate Library for REXX does not need to be shipped and installed with the software program.
- ▶ Maintenance of the Alternate Library for REXX is handled by the z/OS system administrator.

## 1.22 RRS

In z/OS V1R9, RRS now has a batch interface that has commands and parameters to gather the same information that the online interface provides. This implementation allows RRS information to be collected when needed, and then to use this information for problem determination if any failure should occur later.

## 1.23 ISPF

In z/OS R9, ISPF provides the ability to edit ASCII data sets from the ISPF editor directly without converting them to EBCDIC first. Within the ISPF editor, the user may issue the **SOURCE ASCII** command to begin editing in ASCII mode. The **RESET SOURCE** command will revert to EBCDIC (normal) editing mode. Additionally, a **LF** command can be used to “massage” the data, splitting ASCII lines correctly in an FB dataset, or example.

The ISPF editor will also be enhanced to allow files in zFS to be edited, rather than having to use oedit and obrowse. Lastly, a change was made to the ISPF command tables to allow lower case characters to be stored.

Additional ISPF enhancements are listed here:

- ▶ ISPF is enhanced to share profile variables across multiple systems in a Parallel Sysplex. This can eliminate the need for multiple profile data sets in a sysplex.

- ▶ It provides support to use system symbols within data set names when entered in ISPF panels.
- ▶ It offers improved ISPT Edit Undo processing even after the ISPF Edit save command is issued and the data being edited has been saved. Edit undo buffers will be retained by ISPF. This is intended to allow you to remove changes from edited data even after a save command.
- ▶ It provides support for editing and browsing z/OS UNIX and ASCII files.
- ▶ It offers enhanced DSLIST command table support, and REXX variables processing.

## 1.24 Common Information Model

The z/OS V1R9 Common Information Model (CIM) has updated the cross-platform support to a new version of the CIM Schema and the OpenPegasus CIM Server. Along with these updates, the CIM Server is enhanced to register with z/OS Automatic Restart Manager (ARM) and to allow clients to be authenticated through SSL certificates. CIM provides an industry-standard way to externalize information about computing systems so that it can be processed by common tools.

The Common Information Model is a standard data model for describing and accessing systems management data in heterogeneous environments. It allows system administrators or vendors to write applications (CIM monitoring clients) that measure system resources in a network with different operating systems and hardware. With z/OS CIM, it is possible to use the DMTF CIM open standard for systems management which is also implemented on other major server platforms (Linux on zSeries, Linux on xSeries®, i5/OS®, and AIX®).

z/OS CIM implements the CIM server which is based on the OpenPegasus Open Source project. A CIM monitoring client invokes the CIM server that in turn collects z/OS metrics from the system and returns them to the calling client. To obtain the z/OS metrics, the CIM server invokes the z/OS RMF monitoring provider which retrieves the metrics associated with z/OS system resources. The z/OS RMF monitoring provider uses RMF Monitor III performance data.

The metrics obtained by this new API are common across server platforms, so you can use it to create end-to-end monitoring applications.

## 1.25 Metal C runtime library

The XL C Metal compiler option, introduced in z/OS V1R9, generates code that does not have access to the Language Environment support at run time. Instead, the Metal option provides C-language extensions that allow you to specify assembly statements that call system services directly. Using these language extensions, you can provide almost any assembly macro, and your own function prologs and epilogs, to be embedded in the generated HLASM source file. When you understand how the Metal-generated code uses MVS linkage conventions to interact with HLASM code, you can use this capability to write freestanding programs.

Prior to z/OS V1R9, all z/OS XL C compiler-generated code required Language Environment. In addition to depending on the C runtime library functions that are available only with Language Environment, the generated code depended on the establishment of an overall execution context, including the heap storage and dynamic storage areas. These

dependencies prohibit you from using the XL C compiler to generate code that runs in an environment where Language Environment did not exist.

## 1.26 XML System Services

With z/OS V1R9, the XML System Services parser is a follow-on release to the XML System Services component that shipped in z/OS V1R8 and implements performance optimizations, C/C++ interfaces, and functional enhancements allowing the caller to change parser features with a control request. This release also implements support for zAAPs that enhances the cost and performance usage for XML parsing on z/OS.

XML System Services is now very useful for IBM and external exploiters such as ISV and customers because assembler and PL/X skills are becoming increasingly rare. There is widespread availability of C and C++ language development and performance tools that such applications can take advantage of, and many programs are already written in C/C++ and will no longer need to switch into PL/X or assembler to invoke XML system services or make inter-language calls without the benefit of C/C++ header files.

## 1.27 z/OS dbx enhancements

z/OS V1R9 provides z/OS dbx support for IBM WebSphere® Developer Debugger for System z V7.0 (5724-N06) and WebSphere Developer for System z V7.0 (5724-L44). The z/OS dbx enhancements are planned to provide an Eclipse-based graphical user interface (GUI) for interactive, source-level debugging capabilities for compiled System z applications. Running under z/OS UNIX System Services, dbx is designed to enable developers to examine, monitor, and control the running of z/OS UNIX System Services application programs written in C, C++, and High Level Assembler on a z/OS system.

**Note:** The z/OS dbx support for WebSphere Developer Debugger for System z V7.0 and WebSphere Developer for System z V7.0 is planned to be supported on z/OS V1R8 in the second quarter of 2007.

IBM WebSphere Developer for System z V7.0 (5724-L44) includes new support for XL C/C++ mainframe development. The support is available for z/OS V1R8 XL C/C++, whose function can be ordered in the z/OS V1R8 C/C++ Without Debug feature. Core features include the following:

- ▶ XL C/C++ support for development, editing, content assist, enhanced code navigation, and remote syntax checking
- ▶ XL C/C++ builds on MVS or z/OS UNIX System Services
- ▶ Integrated client debugging via Debug Tool
- ▶ Debugging via z/OS dbx
- ▶ Access to z/OS and z/OS UNIX file system resources

## 1.28 Unicode

Unicode is a universal encoding scheme allowing applications to store data regardless of code pages and character sets such as ASCII and EBCDIC. The Unicode services element of

z/OS provides general purpose programming interfaces (APIs) which applications such as DB2® can use to convert data to and from Unicode. The value of storing data in Unicode derives from the ability to store data in any language using the same data server. It enables database consolidation and better interoperability with other platforms (Microsoft® and Java™ applications).

In z/OSV1 R9, Unicode adds function to change iconv() to call Unicode Services. Today, we have two mechanisms for character code conversion: Language Environment's iconv() and Unicode conversion services. iconv() does not use zArchitecture instructions for conversion. Rather than enhance iconv() to use the new instructions, a change is made to use Unicode conversion services. This will make use of the hardware instructions, which will eliminate the need for Language Environment to ship conversion tables, and allow for future investments to be made in a common code base.

TBCS or triple-byte character set is becoming very important to support languages such as Chinese. z/OS currently does not support TBCS. TBCS support will help emerging markets to be competitive. Unicode service upgrade allows customers to use these services at the latest possible levels. New locale support enables customers to use the newest euro locales. The z/OS V1R9 changes are as follows:

- ▶ Along with single, double, quadruple and mixed byte character set, z/OS now supports triple byte character set. This support needs to be added for tables such as CCSID - 964 (EBCDIC Traditional Chinese EUC). “
- ▶ TBCS support will allow CCSID and others to be added in the “to/from CCSIDs” when using Unicode Character Conversion Services.

## Installation considerations

z/OS consists of base elements and optional features. The base elements (or simply *elements*) deliver essential operating system functions. When you order z/OS, you receive all of the base elements. The optional features (or simply *features*) are orderable with z/OS and provide additional operating system functions.

The program number for z/OS Version 1 Release 9 is 5694-A01. When ordering this program number, remember to order all the optional features that you were licensed for in previous releases of z/OS.

In many countries you may order z/OS electronically through ShopzSeries. ShopzSeries provides customers a self-service capability for planning and ordering S/390® software (and service) upgrades over the Web. It is the strategic worldwide self-service ordering system for zSeries software. You can order products through ShopzSeries and have them delivered electronically in some countries.

In most countries, ShopzSeries provides electronic ordering and electronic delivery support for z/OS Service. You can access it directly off the ShopzSeries Web site at:

<http://www.ibm.com/software/shopzseries>

When you order the z/OS product on ServerPac from ShopzSeries, you can choose to have it electronically delivered to you. This electronic ability was made generally available on January 10, 2005.

## 2.1 Ordering z/OS V1R9

Ensure you order the optional priced and unpriced features that you were using in previous releases of z/OS. It is possible to order z/OS V1R9 electronically via ShopzSeries at GA. You can choose to have an electronic ServerPac.

### 2.1.1 Hardware requirements

z/OS V1R9 is planned to run on the following IBM System z servers:

- ▶ z9 BC
- ▶ z9 EC
- ▶ z990
- ▶ z890
- ▶ z900
- ▶ z800

### 2.1.2 Export control features

Remember your export-controlled features, if you desire. Here is the list:

- ▶ z/OS Security Level 3
- ▶ Communications Server Security Level 3

**Note:** There is no longer any ordering considerations for Tivoli NetView® and System Automation since msys for Operations (a former base element) has been removed in z/OS V1R8.

## 2.2 New base elements

z/OS consists of base elements and optional features. The base elements (or simply elements) deliver essential operating system functions. The base elements that are new with z/OS V1R9 are listed in Table 2-1. When you order z/OS, you receive all of the base elements.

Table 2-1 New base elements with z/OS V1R9

New base elements	Description
Alternate Library for REXX	New nonexclusive base element (FMID HWJ9143, JWJ9144 Japanese)
Metal C Runtime Library	New base element (FMID HSD7740)
BCP - new component for future functionality	Component of BCP base element (FMID HPV7740)



## 2.2.1 Alternate Library for REXX

Alternate Library for REXX enables users to run compiled REXX programs. This base element is new in z/OS V1R9. It is nonexclusive to z/OS because the following programs provide equivalent function:

- ▶ The Alternate Library portion of the priced product IBM Library for REXX on zSeries V1R4 (5695-014). The Alternate Library consists of FMIDs HWJ9143 (ENU) and JWJ9144 (JPN). These are the same FMIDs that are now in the z/OS base element.
- ▶ The no-fee Web download Alternate Library for REXX on z/OS.

The fact that the Alternate Library function is now built into z/OS should make the function easier to use.

## 2.2.2 Metal C Runtime Library

The Metal C Runtime Library is a set of LPA-resident C functions that can be called from a C program created using the z/OS XL C compiler Metal option. This base element is new in z/OS V1R9.

As of z/OS V1R9, the BCP also includes Capacity Provisioning (FMID HPV7740) and System REXX for z/OS Base.

## 2.2.3 Elements changed in z/OS V1R9

The function in FMID HBB77SR, System REXX for z/OS base, is integrated in the BCP in z/OS V1R9. It was a Web deliverable on z/OS V1R8 (System REXX Support for z/OS V1R8 and z/OS.e V1R8).

Capacity Provisioning is new in base element BCP with FMID HPV7740.

Java CIM and SLP client is new in base element CIM with FMID JPG290B. Common Information Model (CIM) is a standard data model for describing and accessing systems management data in heterogeneous environments. It allows system administrators to write applications that measure system resources in a network with different operating systems and hardware. To enable z/OS for cross-platform management, a subset of resources and metrics of a z/OS system are mapped into the CIM standard data model. CIM was new in z/OS V1R7.

In z/OS V1R9, the new FMID JPG290B was added to CIM. JPG290B contains a Java programming interface for CIM client applications. Transport layer security (TLS) encryption is performed for CIM by base element Communications Server. CIM does not implement any of its own encryption algorithms.

### Deleted FMIDs

The FMID for the DFSMSdfp English panels, JDZ118E, has been eliminated and the panels have been merged into the base FMID, which is HDZ1190.

## 2.3 Functions withdrawn from z/OS V1R9

The following functions have been withdrawn from z/OS in z/OS V1R9.

## APPC Application Suite

z/OS V1R9 Communications Server discontinues support of the APPC Application Suite. For most of the functions provided by the APPC Application Suite, more full-featured alternative applications exist in modern integrated SNA/IP networks.

## z/OS.e V1R8

The following information is an important consideration when using z/OS.e.

z/OS.e V1R8 (5655-G52) is planned to be the last release of z/OS.e. Marketing, ordering, support, and service for z/OS (5694-A01) remain unaffected.

z/OS.e V1R8 remains orderable until its planned withdrawal from marketing in October 2007. In accordance with the z/OS (5694-A01) and z/OS.e service policy (to provide service support for each release for three years following its general availability date), IBM intends to withdraw service for z/OS.e V1R8 in September 2009.

## 2.4 Functions withdrawn in a future release

IBM plans that z/OS V1R9 will be last release to support English and Japanese ISPF panels in DFSORT™. There will be no replacement for this limited interactive facility. Support for JCL to sort, copy, or merge will continue to be available.

z/OS V1R9 is planned to be the last release of z/OS Communications Server that will support the configuration of Traffic Regulation (TR) policy as part of the Quality of Service discipline. The TR configuration function remains supported, but IBM recommends that you implement it as part of the Intrusion Detection Services (IDS) policy configuration made available in z/OS V1R8.

Note that this change is only for the TR policy *configuration*. The TR policy functions themselves remain unaffected. For more information, refer to the following publications:

- ▶ In *z/OS Communications Server IP Configuration Guide*, see Chapter 16 “Intrusion Detection Services (IDS)”
- ▶ In *z/OS Communication Server IP Configuration Reference*, see Chapter 23 “Intrusion Detection Services (IDS) policy”

### 2.4.1 Changes to driving system requirements

The minimum driving system level for installing z/OS V1R9 is z/OS V1R7 or z/OS.e V1R7. (For installing z/OS V1R8, it was z/OS V1R5 or z/OS.e V1R5.)

If a Customized Offerings Driver (5655-M12) is being used to install z/OS V1R9, the level required is V2.3.1. V2.2 could be used to install z/OS V1R8.

If you are migrating to z/OS V1R8 from z/OS V1R7, or if you will have a different product set than your previous release, you will see increased need for DASD. How much more depends on what levels of products you are running. Keep in mind the DASD required for your z/OS system includes (per the z/OS Policy). That is, it 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 required by other products you might have installed. All sizes include 15% freespace to accommodate the installation of maintenance.

## DASD space requirements

For z/OS V1R9, the total storage required for all the target data sets is 6400 cylinders on a 3390 device. The total storage required for all the distribution data sets listed in the space table is 8900 cylinders on a 3390 device.

The total file system storage is as follows:

- ▶ 2,900 cylinders on a 3390 device for the ROOT file system
- ▶ 50 cylinders for the /etc file system
- ▶ 50 cylinders for the VARWBEM file system (new for CIM element)

The total storage required for the SMP/E SMPLTS is 0 3390 cylinders (there are no load modules in z/OS V1R9 that are both cross-zone and use CALLLIBs, thus the SMPLTS is not needed for permanent storage).

If you are migrating to z/OS V1R9 from a very old operating system release, or if you will have a different product set than your previous release, you will see increased need for DASD space, as shown in Table 2-2; note that sizes are in 3390 cylinders.

Table 2-2 DASD space requirements for installing z/OS V1R9

	z/OS V1R4	z/OS V1R5	z/OS V1R6	z/OS V1 R7	z/OS V1R8	z/OS V1R9
Target	4840	5244	5277	5225	5625	6400
DLIB	6446	6930	7338	7286	7325	8900
HFS	2250	2200	2800	2800	2800	2900

## 2.5 Changed base elements and optional features

There are a number of base elements and optional features that have changed in z/OS V1R9. In z/OS V1R9, the z/OS elements and features are re-structured as follows:

- ▶ Changed base elements:
  - CIM
  - Communications Server
  - Cryptographic Services
  - DFSMSdfp
  - Distributed File Service
  - HCD
  - Integrated Security Services
  - ISPF
  - JES2
  - Language Environment
  - Library Server
  - NFS
  - Run-Time Library Extensions
  - TSO/E
  - z/OS UNIX
- ▶ Changed optional features:
  - C/C++ without Debug Tool
  - Communications Server Security Level 3
  - DFSMSdss™
  - DFSMSHsm
  - DFSMSrmm

- DFSMStvs
- HCM
- JES3
- RMF
- SDSF
- Security Server
- z/OS Security Level 3

## 2.6 Coexistence, fallback, and migration

Prior to z/OS V1R6, four consecutive releases were supported for coexistence, fallback, and migration. Starting with z/OS V1R6, the coexistence, fallback, and migration policy was aligned with the service policy. Because the service policy is a 3-year policy and because z/OS V1R6 was the start of the annual release cycle, three releases will be supported for coexistence, fallback, and migration over a period of three years.

The current policy represents an increase of one year over the two-year period provided by the previous coexistence, fallback, and migration policy of four releases under a six-month release cycle. The intention of the current policy is to simplify and provide greater predictability to aid in release migrations. Therefore, with z/OS V1R9 the following conditions exist; see Figure 2-1.

- ▶ Coexistence of a V1R9 system with a V1R9, V1R8, or V1R7 system is supported.
- ▶ Fallback from a V1R9 system to a V1R8 or V1R7 system is supported.
- ▶ Migration to a V1R9 system from a V1R8 or V1R7 system is supported.

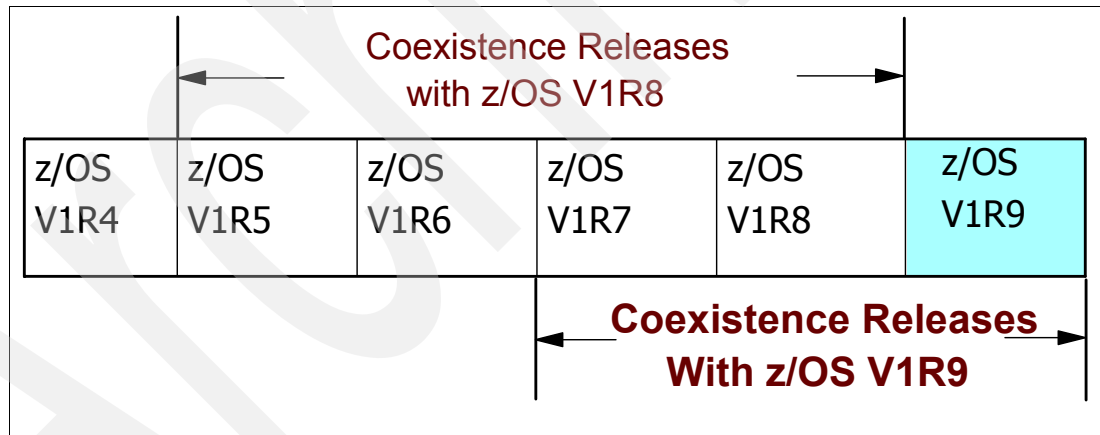


Figure 2-1 Coexistence releases with z/OS V1R9

### z/OS releases and the hardware

Figure 2-2 on page 29 shows all models of the hardware over the current and previous years, and the z/OS releases that used to be supported by the hardware. It also shows which z/OS releases since z/OS Release 4 up to z/OS Release 11 support the System z hardware.

The hardware here also includes the IBM Total Storage products DS8000™, DS6000™, and TS1120. The IBM System Storage™ TS1120 Tape Drive (TS1120 tape drive) offers a solution to address applications that need high capacity, fast access to data or long-term data retention. It is supported in IBM tape libraries, IBM frames that support standalone installation, and in an IBM 3592 Tape Frame Model C20 (3592 C20 frame) attached to a Sun™ StorageTek™ 9310 library.

The tape drive uses IBM 3592 cartridges, which are available in limited capacity (100 GB) for fast access to data, and standard capacity (500 GB) or extended capacity (700 GB), which help to reduce resources to lower total cost. All three cartridges are available in rewritable or Write Once Read Many (WORM) format.

z/OS	G5/G6 Multiprise 3000	z800	z890	z900	z990	z9 EC	z9 BC	DS8000 DS6000	TS1120	End of Service	Coexists with z/OS...	Planned Ship Date
R4	x	x	x <sup>1</sup>	x	x <sup>1</sup>	x <sup>1</sup>	x <sup>1</sup>	x	x <sup>1</sup>	3/07	1.7	
R5	x	x	x	x	x	x	x	x	x	3/07	1.8	
R6		x	x	x	x	x	x	x	x	9/07	1.8	
R7		x	x	x	x	x	x	x	x	9/08*	1.9	
R8		x	x	x	x	x	x	x	x	9/09*	1.10*	
R9		x	x	x	x	x	x	x	x	9/10*	1.11*	9/07
R10*		x	x	x	x	x	x	x	x	9/11*	1.12*	9/08*
R11*		x	x	x	x	x	x	x	x	9/12*	1.13*	9/09*

Figure 2-2 z/OS support summary

## 2.7 54-way support with the z9 EC

z/OS V1R9 has 54-way support on IBM System z9 EC servers. The 54-way support is the sum of CPs, zIIPs, and zAAPs in one z/OS LPAR. The IBM System z9 54-way CPU will be able to process 1 billion transactions per day, which is more than double the performance of its predecessor, the z990. Moreover, reliability improvements in the IBM System z9 design means less customer planned downtime.

z/OS V1R9 is designed to help provide constraint relief, improve overall scalability and performance, and enhance measurement capabilities. It offers new designs to help provide up to 54-way single image support, which includes:

- ▶ Improved SMF data collection and management
- ▶ Improved performance for Coupling Facility (CF) duplexing
- ▶ Global resource serialization (GRS)
- ▶ Couple data set (CDS) I/O
- ▶ Applications using Language Environment heap pools

## 2.8 New address spaces

There are three new address spaces with z/OS V1R9. There is nothing for you to do to start, manage, or stop these new address spaces. However, if you have staff who want to be kept aware of changes to the system, notify them that these address spaces exist:

- ▶ Common event adapter (CEA)

The common event adapter (CEA) provides the ability to deliver z/OS events to C language clients, such as the z/OS CIM server. The CEA address space is started automatically during z/OS initialization and does not terminate.

► **ARCnXXXX**

One of these DFSMSDsss address spaces is started automatically by DFSMSHsm whenever a dump, restore, migration, backup, recover, or CDS backup function is invoked. (A DFSMSDsss address space is not started for recall tasks.) These DFSMSDsss address spaces can reduce the storage used in the DFSMSHsm address space, thus enabling more tasks to be started within the DFSMSHsm address space.

When DFSMSHsm invokes DFSMSDsss through the DFSMSDsss cross-memory application interface, DFSMSHsm requests that DFSMSDsss use a unique address space identifier for each unique DFSMSHsm function and host ID. The address space identifier for each function is in the form ARCnXXXX, where n is a unique DFSMSHsm host ID and XXXX is an abbreviation of a DFSMSHsm function. The abbreviations and corresponding functions are:

- DUMP for dump
- REST for restore
- MIGR for migration
- BACK for backup
- RCVR for recover
- CDSB for CDS backup

For instance, migration for DFSMSHsm host ID 1 would result in a generated address space identifier of ARC1MIGR. The address space terminates automatically when DFSMSHsm terminates.

► **DSSFRDSR**

The purpose of this DFSMSDsss address space is to recover up to 64 data sets concurrently from one or more copy pool backup versions. The address space is started automatically by DFSMSHsm whenever a data set is recovered from DASD using the FRRECOV DSNNAME command. The address space terminates automatically when DFSMSHsm terminates.

## 2.9 System z New Application License Charges (zNALC)

zNALC replaces New Application License Charges (NALC) and z/OS.e, and is to be the IBM strategic z/OS offering for new workloads. zNALC offers a reduced price for z/OS operating system on LPARs where you are running a qualified “new workload” application.

zNALC became available in March 2007. z/OS.e runs only on the z800, z890, z9 BC servers and it now not supported with z/OS V1R9.

zNALC is available only on LPARs where a qualified application is present, among other requirements.

### 2.9.1 zNALC support

zNALC offers a reduced price for the z/OS operating system on LPARs where you are running a qualified new workload application (Qualified Application).

The zNALC offering extends the IBM commitment to sub-capacity pricing, allowing installations with a Qualified Application to obtain a reduced price for z/OS where charges are based on the size of the LPAR(s) executing a Qualified Application, assuming all applicable terms and conditions are met.

## Qualified Applications

The zNALC offering extends the IBM commitment to sub-capacity pricing, allowing customers with a Qualified Application to obtain a reduced price for z/OS where charges are based on the size of the LPAR(s) executing a Qualified Application, assuming all applicable terms and conditions are met.

z/OS with zNALC provides many benefits over previous new workload pricing offers. It provides a strategic pricing model available on the full range of System z servers for simplified application planning and deployment. zNALC provides similar pricing benefits to both z/OS.e pricing and z/OS with NALC pricing. zNALC allows for aggregation across a qualified Parallel Sysplex, which can provide a lower cost for incremental growth across new workloads that span a Parallel Sysplex. zNALC is the IBM strategy, replacing the z/OS.e operating system and the NALC pricing metric.

zNALC is available only on LPARs where a Qualified Application is present, among other requirements. In general, Qualified Applications are those that IBM considers 'new workload,' such as Java language business applications running under WebSphere Application Server, Domino®, SAP, PeopleSoft, or Siebel®.

To implement zNALC, one of the following ways can be selected:

- ▶ Full-Capacity zNALC - charges are based on the full zSeries server capacity where each zNALC product executes
- ▶ Sub-Capacity zNALC - charges are based on the utilization of the LPAR or LPARs where a zNALC product executes

## Hardware requirements

zNALC is available only on IBM z/Architecture® servers (z900, z990, z9 EC, z800, z890, z9 BC, or later) running the z/OS (5694-A01) operating system. z/OS middleware running on the IBM z/Architecture server which qualifies for Workload License Charges (WLC) or Entry Workload License Charges (EWLC) must be priced WLC/EWLC in order for z/OS to be eligible for zNALC charges.

z/OS is eligible for zNALC pricing when running in an LPAR where the Qualified Application is executing. The only other products that may execute in this LPAR are those products that support the Qualified Application. The LPAR must be used exclusively for the Qualified Application and for programs that support the Qualified Application and for no other purpose.

Any logical partition (LPAR) that is designated as a zNALC LPAR must identify itself in one of two ways:

- ▶ By using the naming convention ZNALxxxx, where xxxx may be any letters or numbers.
- ▶ By using the LICENSE=ZNALC IPL parameter. This IPL parameter is available on z/OS Version 1 Release 6 or later systems which have APAR OA20314 applied.

**Note:** zNALC is not available on any server where OS/390 or z/OS.e is licensed or running, or on any server where the pricing metric for z/OS is NALC. Sub-Capacity zNALC is not available on any LPARs where z/OS is running as a guest of z/VM®.

## Criteria to determine which applications are Qualified Applications

An *application* is a computer program that is used to accomplish specific business tasks (such as Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Supply Chain Management (SCM), business information warehouse, accounting, and inventory control programs), including the database server used for that task. In this definition, an application is not a standalone database management system or systems

management tool (that is, related to the management or operation of the computer itself or of other computer programs).

Examples of software that is *not* considered applications are operating system software, database products (except those qualifying as described in section b), transaction managers, tools, utilities, and games.

An application may be considered a Qualified Application if:

- a) It is currently generally commercially available, supported by its manufacturer, and enabled to run under z/OS, and that same Application (with substantially the same functionality) is simultaneously generally commercially available, supported by its manufacturer on, and enabled to run under a UNIX operating system (for example, AIX, HP-UX, Linux, or Solaris™), or Microsoft Windows (collectively, “Distributed Platforms”).
- b) It is a database server running under z/OS and it is operating solely in support of a software program that is currently generally commercially available, supported by its manufacturer, and running in a client/server environment where the business logic (for example, application server) is running on a Distributed Platform.
- c) It is a Java language business application running under WebSphere Application Server (or equivalent). These do not include systems management tools.

IBM will determine whether a particular program is a Qualified Application.

To determine whether other applications can qualify, refer to “How to qualify applications”.

## Examples of Qualified Applications

You may already have an application that has been previously approved. Following are examples of approved Qualified Applications:

- ▶ DB2 for z/OS in support of SAP
- ▶ DB2 for z/OS in support of PeopleSoft
- ▶ DB2 for z/OS in support of Siebel
- ▶ Lotus® Domino

This is not a complete list of Qualified Applications, and more will be added over time.

**Note:** If you already have one of these applications, then simply contact your IBM representative to see what your System z New Application License Charge (zNALC) charges for z/OS will be for that application LPAR.

## How to qualify applications

If your application is not on the list shown in “Examples of Qualified Applications” (for example, a Java language business application running under WebSphere Application Server), then you will need to provide some information in order to get it qualified.

In an e-mail or on company letterhead, describe your application (including the name of the application or workload and a brief explanation of its business purpose) and submit it to IBM for review via your IBM sales representative or IBM Business Partner.



**Note:** If the business application is commercially available from a vendor, you need to supply the vendor name, application Web site, and a short description of the application, including specifically whether it is a commercially available application that is supported on z/OS, UNIX, Microsoft Windows, and/or Linux, and is currently running on z/OS—or whether it is a commercially available application running on z/OS, UNIX, Windows, and/or Linux that accesses data on z/OS (for example, DB2 for z/OS over DRDA® via TCP/IP connection).

## **zNALC and traditional workloads on the same processor**

When z/OS is licensed for both zNALC and non-zNALC (traditional) LPARs on the same machine, IBM will not permit the total billable z/OS MSUs to exceed the reported z/OS peak (the highest simultaneous rolling 4-hour average of all z/OS LPARs). Here are the mechanics for a typical situation:

- ▶ Traditional (WLC/EWLC) MSUs and zNALC MSUs are billed as reported on the SCRT Report.
- ▶ Traditional MSUs + zNALC MSUs exceed the z/OS peak MSUs:
  - Traditional MSUs are billed as reported on the SCRT Report and zNALC MSUs will be reduced such that the sum of the Traditional MSUs plus the zNALC MSUs equals the z/OS peak MSUs.
  - If zNALC MSUs are reduced to the announced 3 MSU minimum, but the sum of the Traditional MSUs plus the zNALC MSUs still exceeds the z/OS peak MSUs, then the Traditional MSUs will be reduced such that the sum of the Traditional MSUs plus the 3 MSUs for zNALC equals the z/OS peak MSUs.

## **Ordering zNALC**

IBM has established a certification process whereby customers must complete a form when they establish zNALC charges, and the form must be renewed each year to maintain zNALC charges. This form requires customers to certify that they meet all the requirements to be eligible for z/OS with zNALC charges. IBM may cancel zNALC charging if a customer fails to submit an annual certification. IBM has the right to audit servers with z/OS with zNALC charges to ensure compliance with all zNALC terms and conditions.

## **2.9.2 NALC users**

Prior to the announcement of zNALC, there was another price metric called New Application License Charges (NALC). In the zNALC announcement, IBM released a Statement of General Direction stating that IBM intends to replace both the z/OS.e operating system and the NALC pricing metric with the zNALC pricing metric, which is available on both IBM z/Architecture high-end and midrange systems.

NALC remains available until withdrawn to customers who dedicate an entire mainframe server to a qualifying e-business workload, such as WebSphere or a qualifying enterprise application workload such as SAP or PeopleSoft. For a product with the NALC pricing metric, there is a single low price per MSU per product and software charges are based upon the capacity of the machine where the product executes. NALC is available to PSLC and WLC customers. NALC provides lower price points for certain features of z/OS, OS/390 and Domino Version 5.

NALC is available on a dedicated e-business mainframe that participates in a Parallel Sysplex environment. Although NALC-priced products are not eligible for aggregation, other non-NALC middleware on the NALC machine may aggregate with middleware across the

Parallel Sysplex environment, if all terms are met. In the case that a machine is dedicated to e-business, and also Sub-Capacity Workload License Charges, then the billable z/OS MSUs and/or Domino NALC MSUs will be based on the values that appear in the monthly Sub-Capacity Reports. This is the only time when IBM terms permit NALC MSUs to be less than full machine-capacity.

For a list of NALC qualifying applications, visit the NALC section of the System z9 and zSeries Software Contracts Web site.

**Note:** With the introduction of zNALC pricing, it is suggested that customers interested in running new workloads on the System z platform no longer rely on NALC or z/OS.e but rather upon zNALC to obtain reduced price points for environments with new workload applications.

### 2.9.3 zNALC and SCRT and APAR OA20314

In July 2007, IBM made Sub-Capacity Reporting Tool (SCRT) Version 14 Release 1.0 available for download. In order to take advantage of the new SCRT V14.1.0 support for System z New Application License Charges (zNALC) in an LPAR which does not use the ZNALxxxx naming convention, customers must also have the BCP APAR OA20314 applied to their z/OS system prior to collecting SMF data.

#### IEASYSxx parmlib member

The IEASYSxx parmlib member LICENSE parameter is now enhanced to allow LICENSE=ZNALC to be specified as the licensed environment. With LICENSE=ZNALC specified, there is no longer a requirement for LPARs to be named in the form ZNALxxxx in order to qualify for zNALC subcapacity pricing. The use of LICENSE=ZNALC is not a requirement, however, and customers can still opt to use the LPAR name in the form ZNALCxxx to also take advantage of zNALC subcapacity pricing.

#### SMF record type 89

Along with the enhanced LICENSE parameter, the SMF type 89 record is updated by this APAR, with a new bit indicator for recording when the system is IPLed with zNALC subcapacity pricing requested. This support also adds a new field to the SMF type 89 record, to record the LPAR name when z/OS is run as a VM guest.

#### D IPLINFO command

The D IPLINFO command was also enhanced to show LICENSE=ZNALC when a system was IPLed with zNALC pricing requested, whether via ZNALxxxx LPAR name or by the use of the LICENSE=ZNALC system parameter.

This APAR allows customers to use a LICENSE=ZNALC IPL parameter in place of the zNALC LPAR naming convention.

**Note:** SCRT V14.1.0 is the first version of the program to support zNALC customers who wish to exploit the new LICENSE=ZNALC parameter made possible through z/OS APAR OA20314. The other changes coming with V14.1.0 are listed on the Web at:

[http://ibm.com/zseries/swprice/scrt/scrt\\_new.html](http://ibm.com/zseries/swprice/scrt/scrt_new.html)

## Coupling Facility enhancements

With z/OS v1R9, there is improved duplexing performance using the new CFLEVEL=15 to CFLEVEL=15 duplexing. With this new support, you can improve duplexing performance significantly by duplexing between two Coupling Facilities at CFLEVEL=15 or higher on an IBM System z9 109 (z9-109) server. However, the system will only optimize performance between two CFLEVEL=15 Coupling Facilities when there are no path busy or other delay conditions affecting requests to the Coupling Facility.

New support is added to put a Coupling Facility into maintenance mode. In preparation for removing a Coupling Facility from the sysplex, you can place it in maintenance mode. This step prevents systems from allocating any structures in that Coupling Facility while you prepare to take it down for upgrade procedures. Operator commands can move structures from a Coupling Facility that is to be placed into maintenance mode to another Coupling Facility.

In this chapter we describe the following enhancements to the Coupling Facility in z/OS V1R9:

- ▶ CF duplexing performance, accounting and measurement enhancements
- ▶ CF maintenance mode
- ▶ CFCC level 15

## 3.1 CF duplexing performance enhancements

System-managed duplexing rebuild allows the system to allocate another structure in a different Coupling Facility for the purpose of duplexing the data in the structure. It was introduced in z/OS V1R2 and provides a recovery mechanism in a Parallel Sysplex supplying availability to the environment.

**Information:** System-managed duplexing rebuild is a process managed by the system that allows a structure to be maintained as a duplexed pair. The process is controlled by CFRM policy definition as well as by the subsystem or exploiter owning the structure. The process can be initiated by operator command (**SETXCF**) or programming interface (**IXLREBLD**), or it can be MVS-initiated.

The new control facility control code (CFCC) provides support to streamline the CF-to-CF synchronization protocols currently involved in CF duplexing. This enhancement reduces the overhead of CF duplexing, and may help make duplexing a more viable alternative for use in providing high availability for CF Structure data.

**Note:** The support applies to system-managed duplexing.

The new CFCC also provides additional CF measurement information to provide CF processor and CF structure execution time for enhanced accounting of CF and CF structure utilization.

This CFCC support is associated with a new CFLEVEL 15. This enhancement reduced the overhead of CF duplexing, improved performance (service time) and throughput for duplexed requests, and high availability for CF structure data. With this new support, duplexing performance is improved significantly by duplexing between two Coupling Facilities at CFLEVEL=15 or higher on IBM System z9 processors.

The CFCC support includes enhancements in the following support:

- ▶ There are CF duplexing performance enhancements that provide an internal protocol improvement for performance benefit. There are no new exploiter changes associated with this support.
- ▶ There are XES Coupling Facility measurement enhancements that can be seen in RMF reports. The additional CF measurement extensions are exposed through the IXLMG interface on the IXLYAMDA accounting and measurement data area. New information is added to the IXLYAMDA data area mapping for Coupling Facility, and for Coupling Facility structure information, allowing RMF to display the following:
  - Dynamic CF dispatch indication to show whether or not dynamic CF dispatching is active for the CF
  - Counts of shared and dedicated CF processors to indicate the number of shared or dedicated processors in the Coupling Facility
  - CF processor weights indicate the weight assigned to the processor
  - CF structure execution time specifies the total number of microseconds that any processor is in command execution or in execution of a background process for the particular list/lock structure, and specifies the total number of microseconds that any processor is in command execution or in execution of a background process for the particular cache structure

**Coexistence:** The CF duplexing performance enhancements will only apply if a z/OS V1R9 system (or downlevel system with the support) allocates the structure. A structure allocated to take advantage of the CF duplexing enhancements can be used by, and can coexist with, z/OS systems without the support.

The support is included in z/OS V1R9 and is available via a PTF on lower level releases (V1R6 through V1R8) via APAR OA17055.

### 3.1.1 CFLEVEL 15

The control facility control code (CFCC) level 15 contains CFCC multitasking enhancements to provide an increase in the number of supported CF tasks, from 48 to 112. If you are migrating to a new CFCC level, you have to make appropriate Coupling Facility structure size updates in the z/OS Coupling Facility resource management (CFRM) policy. These new multitasking enhancements provide improved CF performance and throughput in Parallel Sysplex configurations, as follows:

- ▶ The enhancements allow many coupling links and subchannels, and therefore allow many concurrent CF operations to execute in the CF.
- ▶ They use system-managed CF structure duplexing as a high availability mechanism for CF structure data.
- ▶ They operate at extended distances between the z/OS systems and the Coupling Facility, such as in GDPS® configurations.

#### Migration to CFLEVEL 15

When migrating to CFLEVEL 15 from earlier CFLEVELs, the multitasking enhancements will create significant growth in the size of many CF structures. If the structures are not appropriately resized to allow for this growth, then problems or outages may result from an unexpected reduction in the number of usable structure objects in a CF structure.

Before you install CFLEVEL 15, you must plan for this structure size growth. It is possible that structures previously usable with a given structure size may not even be able to be allocated in a CFLEVEL 15 with the same structure size.

When migrating to a new CFCC level, run the Coupling Facility Structure Sizer (CFSIZER) tool. This tool sizes structures, taking into account the amount of space needed for the current CFCC levels. The tool sizes for the most currently available level, and you may find that the results are oversized if you use an earlier CFCC level. You can find the CFSIZER tool at:

<http://www.ibm.com/servers/eserver/zseries/cfsizer/>

Then, you can make the corresponding structure size updates to the CFRM policies and activate the updated CFRM policy to be used in the sysplex.

**Note:** To support migration from one Coupling Facility level to the next, you can run different levels of the Coupling Facility concurrently as long as the Coupling Facility LPs are running on *different* processors.

CF LPs running on the same processor share the same Coupling Facility control code EC level. A single processor cannot support multiple Coupling Facility levels.

## Structure size growth

When you are migrating CF levels, then you may have to increase lock, list, and cache structure sizes in order to support the new function. This adjustment can impact the system when it allocates structures or copies structures from one Coupling Facility to another at different CF levels. The Coupling Facility structure sizer tool is designed to size structures for you, and takes into account the amount of space needed for the current CFCC levels.

The amount of per-structure size growth for a variety of CF structure types, when migrating from CFLEVEL 14 to 15, is a real consideration. Growth may be somewhat larger if migrating to CFLEVEL 15 from a CFLEVEL prior to 14. The expected amount of CF structure growth is a fixed and absolute amount per structure, not a percentage increase based on the current allocated structure size. Therefore, the current size of the structure is not a factor in determining the amount of the increase. Instead, the size increase is a function of the maximum data entry size supported by the particular CF structure exploiters.

**Attention:** The amount of storage in CF images may need to be increased to accommodate this growth. Also evaluate the need to provide an adequate amount of unused CF storage space for recovery in the event of the loss of a CF image.

## Hardware requirements

The hardware required to support the CFCC level 15 are the z9 EC or z9 BC with support element and HMC version 2.9.2 plus MCLs.

## Potential problems with CFLEVEL 15 installed

The enhancements for CF duplexing available with CFLEVEL 15 are designed to improve performance and this performance improvement can be affected and not realized when the following conditions occur:

- ▶ Path busy conditions

When a sysplex experiences a significant level of path busy conditions, you will rarely be able to exploit the performance enhancements possible in a CFLEVEL=15 to CFLEVEL=15 duplexing configuration.

**Note:** In this case, you should consider upgrading your configuration to increase Coupling Facility link capacity. For example, you might use dedicated Coupling Facility links (rather than shared) or provide additional shared Coupling Facility links to resolve the path busy conditions.

- ▶ Heavily used Coupling Facilities

When installed at CFLEVEL=15 and the Coupling Facilities are used heavily or overused, to the extent that it causes some delay of requests to the Coupling Facility, the system will not be able to exploit the performance improvement.

**Note:** In this case, you should consider upgrading your configuration to include additional Coupling Facility capacity. For example, you might add more processors, use dedicated processors, or turn off dynamic dispatching to add more Coupling Facility capacity.

## 3.2 CF measurement enhancements

The enhancement of streamlining the processing in the CF duplexing protocols is expected to yield a measurable performance improvement to the duplexed CF service time—and this is an elapsed time benefit. The benefit is based on the configuration and the specific structure, depending on how many of the requests are duplexed. Savings may be quite significant at extended distances.

Additional Coupling Facility measurement extensions have been added to provide CF processor and CF structure execution time for enhanced accounting of CF and CF structure utilization. These enhancements allow for:

- ▶ Better accounting of the processor utilization of the CFs and CF structures
- ▶ Better tuning and capacity planning for CF processor resources as a result of the more granular information
- ▶ Improved CF tuning and capacity planning on a structure basis

### D CF command

The **D CF** command output is enhanced to report on the following new information:

- ▶ The number of shared and dedicated processors in the Coupling Facility
- ▶ The dynamic CF dispatching setting for the Coupling Facility

This information can be displayed from any z/OS system with the software support installed, for every Coupling Facility that is at CFLEVEL 15 that contains this support, and is connected and managed. You can view this information in the **DISPLAY CF** command output, as shown in Figure 3-1.

```
-D CF
IXL150I 11.43.35 DISPLAY CF 762
COUPLING FACILITY 002094.IBM.02.00000002991E
                PARTITION: OF CPCID: 00
                CONTROL UNIT ID: FFF5

NAMED CF1
COUPLING FACILITY SPACE UTILIZATION
ALLOCATED SPACE          DUMP SPACE UTILIZATION
STRUCTURES:             184832 K          STRUCTURE DUMP TABLES:           0 K
DUMP SPACE:             2048 K          TABLE COUNT:                     0
FREE SPACE:             770560 K        FREE DUMP SPACE:                   2048 K
TOTAL SPACE:            957440 K        TOTAL DUMP SPACE:                   2048 K
                                MAX REQUESTED DUMP SPACE:           0 K
VOLATILE:                YES            STORAGE INCREMENT SIZE:           512 K
CFLEVEL:                 15
CFCC RELEASE 15.00, SERVICE LEVEL 00.18
BUILT ON 03/26/2007 AT 12:25:00
COUPLING FACILITY HAS 1 SHARED AND 0 DEDICATED PROCESSORS
DYNAMIC CF DISPATCHING: ON
```

Figure 3-1 Display CF command output

**Note:** Dynamic CF dispatching does not apply to dedicated processors.

The Coupling Facility processor information included is:

- ▶ Indication of whether the processor is shared or dedicated
- ▶ Indication of the weight assigned to the processor

The Coupling Facility structure information included is:

- ▶ The percentage of the total Coupling Facility CPU (execution time) that was consumed by each structure in the CF during a given interval of time

**Note:** These values are only reset when the structure is initially allocated. This support is associated with the Coupling Facility control code (CFCC) CFLEVEL 15, which is exclusive for System z9.

### 3.2.1 RMF enhancements

By exploiting CF level 15 (CFLEVEL 15), RMF provides additional data in the Monitor III Coupling Facility reports and the Postprocessor Coupling Facility Activity report.

For example, the reports display CF utilization per structure and whether dynamic CF dispatching is turned on, as shown in Figure 3-1 on page 39.

With the extended **D** CF command architecture that is introduced by a CFLEVEL 15, new support for granular CF processor utilization accounting is implemented in addition to other minor accounting and measurement extensions.

The processor busy time is provided not only for the CF as a whole, but on a per-structure basis. In addition to this the CF dynamic dispatching setting, the number of shared and dedicated processors, and the subchannel busy value is integrated in the existing the RMF CF reports to improve the CF diagnosis possibilities.

RMF will report the new CF metrics only if the extended CF command architecture is available. On systems running pre-z/OS V1R9, XCF APAR OA17055 has to be installed.

In addition, RMF provides new overview conditions for the post processor based on SMF record 74-4.

**Note:** The RMF support is included in z/OS V1R9 and is available via a PTF on lower level releases (V1R6 through V1R8) via APAR OA17070.

#### **RMF post processor Coupling Facility Activity report**

The RMF post processor Coupling Facility Activity report is extended by including the following information, which is shown in Figure 3-2 on page 41:

- ▶ CF processor utilization by structure
- ▶ Dynamic CF dispatching status
- ▶ Number of dedicated or shared processors
- ▶ Average weight of shared processors

The structure summary section of the Coupling Facility Activity Usage summary groups the structure summary data by structure type. The values shown for each structure are extended by % of CF utilization, which shows the structure-related processor busy time for an allocated structure compared to the total CF processor busy time.



The calculation is shown as follows:

$$\% \text{ OF CF UTIL} = \frac{\text{Structure execution time}}{\text{CF busy time}} * 100$$

**Note:** The new column % OF CF UTIL sums to less than 100%, which is expected, because not all CF processor time is attributable to structures. N/A is shown in this field if the required CF level is missing or the software prerequisite (XCF APAR OA17055 or z/OS V1R9) is not installed.

Figure 3-2 and Figure 3-3 on page 42 show the new fields.

COUPLING FACILITY USAGE SUMMARY												
STRUCTURE SUMMARY												
TYPE	STRUCTURE NAME	STATUS CHG	ALLOC SIZE	% OF CF STOR	# REQ	% OF ALL REQ	% OF CF UTIL	AVG REQ/ SEC	LST/DIR ENTRIES TOT/CUR	DATA ELEMENTS TOT/CUR	LOCK ENTRIES TOT/CUR	DIR REC DIR XI'S
LIST	DB8FU_SCA	ACTIVE	9M	0.9	0	0.0	6.2	0.00	6464	13K	N/A	N/A
									112	425	N/A	N/A
	ISTMNPS	ACTIVE	13M	1.3	0	0.0	0.0	0.00	4163	8308	N/A	N/A
									1	0	N/A	N/A
	IXC_DEFAULT_1	ACTIVE	32M	3.4	19953	11.5	12.0	2.77	5767	5747	N/A	N/A
									1	16	N/A	N/A
	IXC_DEFAULT_3	ACTIVE	17M	1.8	2243	1.3	3.6	0.31	2061	2047	N/A	N/A
									1	32	N/A	N/A
	RRS_DELAYEDUR_1	ACTIVE	13M	1.3	2156	1.2	0.9	0.30	3525	10K	N/A	N/A
									6	48	N/A	N/A
	RRS_MAINUR_1	ACTIVE	13M	1.3	2156	1.2	1.6	0.30	2933	12K	N/A	N/A
									6	48	N/A	N/A
	RRS_RESTART_1	ACTIVE	13M	1.3	1330	0.8	0.3	0.18	6729	6730	N/A	N/A
									5	32	N/A	N/A
	RRS_RMDATA_1	ACTIVE	13M	1.3	31190	18.0	6.2	4.33	19K	19K	N/A	N/A
									5	32	N/A	N/A
	SYSTEM_LOGREC	ACTIVE PRIM	17M	1.8	8	0.0	0.0	0.00	2027	16K	N/A	N/A
		ACTIVE SEC							68	329	N/A	N/A
	SYSTEM_OPERLOG	ACTIVE	33M	3.5	817	0.5	0.8	0.11	29K	29K	N/A	N/A
									20K	22K	N/A	N/A
LOCK	DB8FU_LOCK1	ACTIVE	8M	0.9	0	0.0	2.4	0.00	11K	0	2097K	N/A
									0	0	0	N/A
	IGWLOCK00	ACTIVE	14M	1.5	0	0.0	39.3	0.00	33K	0	2097K	N/A
									0	0	21	N/A
	ISGLOCK	ACTIVE	9M	0.9	113494	65.5	9.3	15.76	0	0	1049K	N/A
									0	0	2805	N/A
CACHE	SYSIGGCAS_ECS	ACTIVE	5M	0.5	0	0.0	0.2	0.00	980	970	N/A	0
									0	0	N/A	0
	SYSZWLM_WORKUNIT	ACTIVE	12M	1.3	0	0.0	0.2	0.00	1052	2093	N/A	0
									0	0	N/A	0
STRUCTURE TOTALS			217M	23.2	173347	100	83.0	24.08				

Figure 3-2 Coupling Facility Activity report - Structure Summary

PROCESSOR SUMMARY							
COUPLING FACILITY	2094	MODEL S18	CFLEVEL 15	DYNDISP	ON		
AVERAGE CF UTILIZATION (% BUSY)	0.0	LOGICAL PROCESSORS:	DEFINED	1	EFFECTIVE	1.0	
			SHARED	1	AVG WEIGHT	10.0	

Figure 3-3 Coupling Facility Activity report - Processor Summary section

**Note:** N/A is shown in the % OF CF UTIL column if the required CF level is missing or the software prerequisite (XCF APAR OA17055 or z/OS V1R9) is not installed.

### 3.3 RMF Monitor III Data Portal for z/OS

The z/OS RMF Distributed Data Server (DDS) provides a Web front-end to sysplex-wide RMF Monitor III online performance data. The performance data with z/OZ V1R9 is enhanced to display CF duplexing information.

Using a Web browser that can display XML documents using XSL style sheets (like Mozilla 1.4 or above, Netscape 7.0 or above, or Microsoft Internet Explorer® 5.5 or above), it provides an easy-to-use interface to RMF online performance monitoring data.

#### Starting the Distributed Data Server

If RMF Monitor III is up and running, simply use an MVS console to start the RMF Distributed Data Server on exactly one system in the sysplex by entering the following command:

```
START GPMSERVE
```

#### Starting the Web browser interface

Open your favorite Web browser and type the following URL into the location bar:

```
http://<yourhost ip address>:8803/
```

The DDS is running on <yourhost> and uses default DDS TCP/IP port 8803. The port number and other settings of the DDS can be configured in GPMSRVxx PARMLIB member. The first display is the RMF Monitor III Data Portal for z/OS home page, as shown in Figure 3-4 on page 43.



Figure 3-4 RMF Monitor III Data Portal for z/OS home page

### New RMF reports

Figure 3-5 on page 44 shows the new RMF reports with z/OS V1R9 that display the following:

- ▶ XCFGROUP (XCF Group Statistics)
- ▶ XCFOVW (XCF Systems Overview)
- ▶ XCFPATH (XCF Path Statistics)
- ▶ XCFSYS (XCF System Statistics)

### RMF Monitor III Coupling Facility Overview report

The RMF Monitor III Coupling Facility Overview report, CFOVER as shown in Figure 3-5 on page 44, is enhanced to provide the following new information:

- ▶ Dynamic CF dispatching setting
- ▶ Number of dedicated processors
- ▶ Number of shared processors
- ▶ Average weight of shared processors

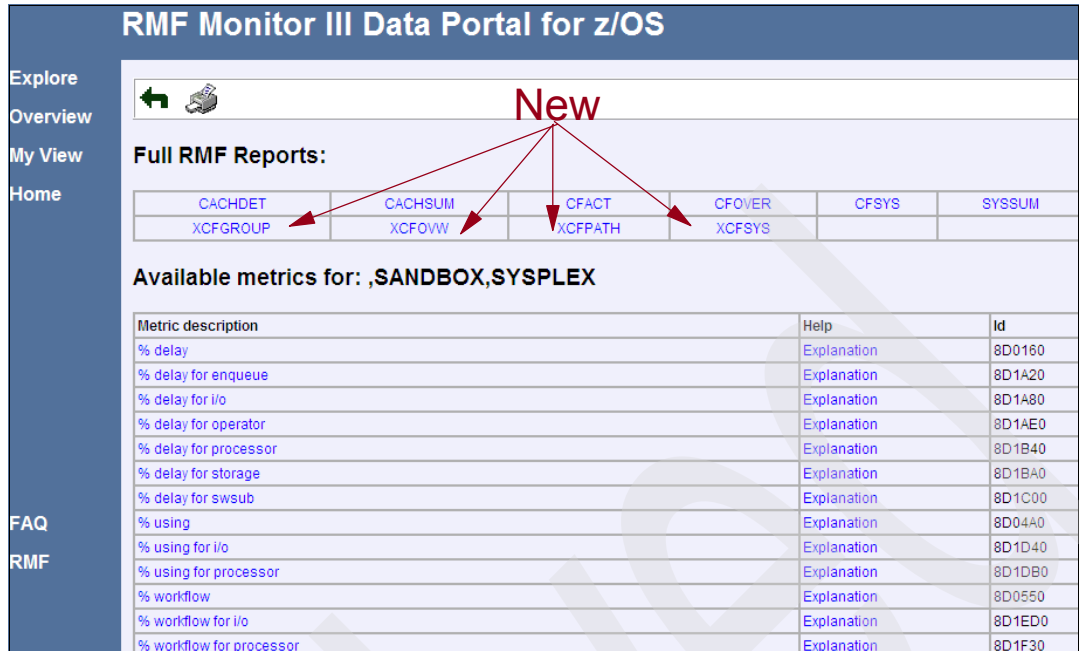


Figure 3-5 Panel that displays the full RMF reports showing new reports

You can see this new information in Figure 3-6.

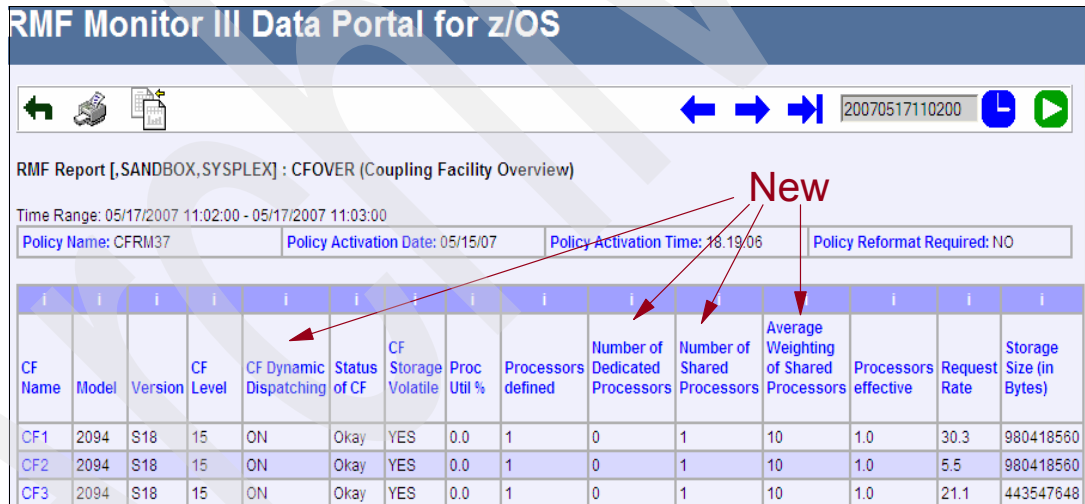


Figure 3-6 Part of Coupling Facility Overview report

### Coupling Facility Activity (CFACT) report

You select the RMF Monitor III Coupling Facility Activity report by clicking CFACT, as shown in Figure 3-5. CFACT report is enhanced to show the following new fields in Figure 3-7 on page 45:

- ▶ Structure status
- ▶ CF utilization percentage of an active structure

This is the processor utilization percentage for an allocated structure. The structure execution time is related to the total CF-wide processor busy time. The sum of the values in this column is less than 100%, because not all CF processor time is attributable to structures.

N/A is shown in this field if the CF level is lower than 15.

**RMF Monitor III Data Portal for z/OS**

RMF Report [,SANDBOX,SYSPLEX] : CFACT (Coupling Facility Activity)

Time Range: 05/16/2007 14:16:00 - 05/16/2007 14:17:00

Structure Name	Structure Type	Structure Status	Extended Structure Status	System Name	CF Utilization %	Structure Execution %	Sync Rate	Sync Avg Service Time	Sync Request Count	Async Rate	Async Avg Service Time	Asyn Requ Cour
DB8FU_LOCK1	LOCK	A	ActivePersistent	*ALL	2.6	0.0	0.0	0	0	0.0	0	0
	LOCK			SC65			0.0	0	0	0.0	0	0
	LOCK			SC70			0.0	0	0	0.0	0	0
DB8FU_SCA	LIST	A	ActivePersistent	*ALL	6.8	0.0	0.0	0	0	0.0	0	0
	LIST			SC65			0.0	0	0	0.0	0	0
	LIST			SC70			0.0	0	0	0.0	0	0
EJESGDS_WTSCPLX4	LIST	A	ActivePersistent	*ALL	0.1	0.0	0.0	0	0	0.0	0	0
	LIST			SC65			0.0	0	0	0.0	0	0
	LIST			SC70			0.0	0	0	0.0	0	0
IGWLOCK00	LOCK	A	ActivePersistent	*ALL	46.4	0.0	0.0	0	0	0.0	0	0
	LOCK			SC65			0.0	0	0	0.0	0	0

Figure 3-7 Part of Coupling Facility Activity report

The RMF Monitor III Coupling Facility Systems View report, report CFSYS in Figure 3-5 on page 44, is enhanced to display the following:

- ▶ Subchannel Delay %  
This is the percentage of all Coupling Facility requests that z/OS had to delay because it found all Coupling Facility subchannels busy.
- ▶ Subchannel busy value  
This is the percentage of the Coupling Facility subchannel utilization. This value is calculated from the sum of synchronous and asynchronous Coupling Facility request times related to the MINTIME.

CF Name	System Name	Subchannel Delay %	Subchannel Busy %	Paths Available	Paths Delay %	Sync Rate	Sync Avg Service Time	Async Rate	Async Avg Service Time	Async Changed %	Async Delay %	Total Request Count	Sync Request Count	Async Request Count	Sync to Async Conversion Count
CF1	SC65	0.0	0.1	3	0.0	1.3	350	9.8	1535	0.0	0.0	1055	78	585	0
	SC70	0.0	0.1	3	0.0	1.5	232	6.9	1669	0.0	0.0	901	89	411	0
CF2	SC65	0.0	0.0	2	0.0	0.0	0	1.1	1705	0.0	0.0	189	0	67	0
	SC70	0.0	0.0	2	0.0	0.0	0	2.4	705	0.0	0.0	269	0	146	0
CF3	SC65	0.0	0.1	2	0.0	2.0	310	15.0	932	0.0	0.0	1126	117	897	0
	SC70	0.0	0.1	2	0.0	2.7	337	13.9	1247	0.0	0.0	1163	164	832	0

Figure 3-8 Part of Coupling Facility Systems View report

### 3.4 CF maintenance mode

z/OS V1R9 provides an easier way to prepare Coupling Facilities for maintenance. New support includes placing Coupling Facilities into a new state, called *maintenance mode*. CF maintenance mode provides a mechanism to prevent usage of the CF for structure allocation, and provides an improvement to the process of moving CF structures out of Coupling Facilities in preparation for maintenance.

**Important:** A CF may only be placed into and out of maintenance mode by a system running z/OS V1R9.

The XCF allocation algorithm eliminates it from the list of Coupling Facilities that are eligible for structure allocation (without any CFRM policy update).

The new maintenance mode support can greatly simplify operational procedures related to taking down a CF for maintenance or upgrade in a Parallel Sysplex. In particular, the need to laboriously update or maintain several alternate copies of the CFRM policy that omit a particular CF which is to be removed for maintenance is avoided.

A subsequent rebuild or **REALLOCATE** command processing will also remove any CF structure instances that were already allocated in that CF at the time it was placed into maintenance mode. In conjunction with the **REALLOCATE** command, the new maintenance mode support can greatly simplify operational procedures related to taking down a CF for maintenance or upgrade in a Parallel Sysplex.

This enhancement can eliminate some of the complexity of managing Coupling Facilities in preparation for a maintenance or upgrade action. For CFRM policy updates, removing the CF or maintaining multiple CFRM policies can be avoided.

### 3.4.1 Migration and coexistence

A Coupling Facility may only be placed into and out of maintenance mode by a system running z/OS V1R9. z/OS systems running V1R8 down to V1R6 with APAR OA17685 installed can *recognize* that a CF is in maintenance mode and is not eligible for CF structure allocation. But the down-level systems cannot place a CF into or out of maintenance mode, even with the APAR installed.

z/OS systems running V1R8 down to V1R6 with this support installed can recognize that a CF is not eligible for CF structure allocation, and messages IXC361I, IXC362I, IXC367I, IXL015I, IXC574I, and IXC463I have been modified with a new message insert indicating ALLOCATION NOT PERMITTED.

**Important:** A sysplex that is falling back to a configuration without any z/OS V1R9 systems may leave a CF stuck in maintenance mode. Before any fallback actions are taken, ensure that all the CFs are taken out of maintenance mode. A CF maintenance mode indication will be cleared by a sysplex-wide IPL.

### 3.4.2 Using the CF maintenance mode

A new **SETXCF** command is provided that allows a Coupling Facility to be placed into and taken out of maintenance mode. You place the CF that will be removed from the environment into maintenance mode by using the following command:

```
SETXCF START,MAINTMODE,CFNAME=(cfname1,[cfname2,]...)
```

#### Place a CF into maintenance mode

Following is a typical procedure to place a particular Coupling Facility into maintenance mode.

1. Place CF1 into maintenance mode.

```
-SETXCF START,MAINTMODE,CFNAME=CF1
IXC369I THE SETXCF START MAINTMODE REQUEST FOR COUPLING FACILITY
CF1 WAS SUCCESSFUL.
```

Figure 3-9 Start maintenance mode command

When a Coupling Facility is placed in maintenance mode, it is ineligible for CF structure allocation purposes. This step prevents systems from allocating any structures in this CF.

Use the **D XCF,CF,CFNM=cfname** command to confirm that the CF went in maintenance mode, as shown in Figure 3-10 on page 48. This figure shows that CF1 is in maintenance mode, but it still has allocated structures. Two other Coupling Facilities (CF2 and CF3) are shown, and are not in maintenance mode.

```

-D XCF,CF,CFNM=CF1
IXC362I 15.54.43 DISPLAY XCF 869
CFNAME: CF1
  ALLOCATION NOT PERMITTED
  MAINTENANCE MODE
  STRUCTURES:
DB8FU_LOCK1          DB8FU_SCA          IGWLOCK00
ISGLOCK              ISTMNPS             IXC_DEFAULT_1
IXC_DEFAULT_3        RRS_DELAYEDUR_1    RRS_MAINUR_1
RRS_RESTART_1        RRS_RMDATA_1        SYSIGGCAS_ECS
SYSTEM_LOGREC(OLD)   SYSTEM_OPERLOG(NEW) SYSZWLM_WORKUNIT

-D XCF,CF,CFNM=CF2
IXC362I 15.47.48 DISPLAY XCF 862
CFNAME: CF2
  STRUCTURES:
EJESGDS_WTSCPLX4     IXC_DEFAULT_2       IXC_DEFAULT_4
SYSARC_PLEXO_RCL     SYSTEM_OPERLOG(OLD)

-D XCF,CF,CFNM=CF3
IXC362I 15.47.53 DISPLAY XCF 864
CFNAME: CF3

  STRUCTURES:
  ISTGENERIC          SYSTEM_LOGREC(NEW)   SYSZWLM_991E2094

```

Figure 3-10 Display of CF1, CF2, and CF3

## 2. Use the **REALLOCATE** command.

Next, use the **SETXCF START, REALLOCATE** command to evaluate and process the CF structures. This will move the CF structures out of the CF that has been placed into maintenance mode. This will not move any CF structures into the CF that has been placed into maintenance mode.

**Note:** The existing **REALLOCATE** process uses the XCF allocation algorithm, which has changed in support of maintenance mode. It will view a CF that is in maintenance mode as being ineligible for structure allocation purposes. It will also view a CF in maintenance mode as an undesirable location, so that it will serially relocate structures out of CFs that are in maintenance mode as part of its normal processing.

The new maintenance mode support, in conjunction with the **REALLOCATE** process, provides a simpler way to prepare their Coupling Facilities for maintenance.

The CF that has been placed into maintenance mode is now empty and ready for upgrade action or maintenance. Any new CF structure allocation will also avoid placing structures in a CF that is in maintenance mode because when a Coupling Facility is placed in maintenance mode it is ineligible for CF structure allocation purposes.

Part of the **REALLOCATE** command output was extracted, as shown in Figure 3-11 on page 49. Note that **REALLOCATE** processes each structure sequentially.



```

-SETXCF START,REALLOCATE
IXC543I THE REQUESTED START,REALLOCATE WAS ACCEPTED.
IXC521I REBUILD FOR STRUCTURE IGWLOCK00
HAS BEEN STARTED
IXC526I STRUCTURE IGWLOCK00 IS REBUILDING FROM
COUPLING FACILITY CF1 TO COUPLING FACILITY CF2.
REBUILD START REASON: OPERATOR INITIATED
INFO108: 00000028 00000028.
IXC521I REBUILD FOR STRUCTURE IGWLOCK00
HAS BEEN COMPLETED
IXC521I REBUILD FOR STRUCTURE ISGLOCK
HAS BEEN STARTED...
...
IXC544I REALLOCATE PROCESSING FOR STRUCTURE IXC_DEFAULT_4
WAS NOT ATTEMPTED BECAUSE
STRUCTURE IS ALLOCATED IN PREFERRED CF
IXC545I REALLOCATE PROCESSING RESULTED IN THE FOLLOWING:
    13 STRUCTURE(S) REALLOCATED - SIMPLEX
     2 STRUCTURE(S) REALLOCATED - DUPLEXED
     0 STRUCTURE(S) POLICY CHANGE MADE - SIMPLEX
     0 STRUCTURE(S) POLICY CHANGE MADE - DUPLEXED
     6 STRUCTURE(S) ALREADY ALLOCATED IN PREFERRED CF - SIMPLEX
     0 STRUCTURE(S) ALREADY ALLOCATED IN PREFERRED CF - DUPLEXED
     0 STRUCTURE(S) NOT PROCESSED
    59 STRUCTURE(S) NOT ALLOCATED
    20 STRUCTURE(S) NOT DEFINED
-----
    100 TOTAL

     0 ERROR(S) ENCOUNTERED DURING PROCESSING
IXC543I THE REQUESTED START,REALLOCATE WAS COMPLETED.

```

Figure 3-11 Part of REALLOCATE command output

**Note:** Enhancements for the REALLOCATE process are provided by APAR OA08688.

The REALLOCATE process removes all structures from a CF and reallocates the structures on the other CFs in the environment as defined in CFRM policy. When it finishes, messages IXC545I and IXC543I are issued to the operator.

An empty CF in maintenance mode is ready to be removed from a sysplex. As shown in Figure 3-12 on page 50, CF1 is empty and ready to be removed from the sysplex for maintenance and all structures are allocated in CF2 and CF3.

```

-D XCF,CF,CFNM=CF1
IXC362I 15.57.54 DISPLAY XCF 989
CFNAME: CF1
ALLOCATION NOT PERMITTED
      MAINTENANCE MODE

      NO STRUCTURES ARE IN USE BY THIS SYSPLEX IN THIS COUPLING FACILITY

-D XCF,CF,CFNM=CF2
IXC362I 15.57.59 DISPLAY XCF 993
CFNAME: CF2
STRUCTURES:
DB8FU_LOCK1          DB8FU_SCA          EJESGDS_WTSCPLX4
IGWLOCK00           ISGLOCK           ISTMNPS
IXC_DEFAULT_1       IXC_DEFAULT_2       IXC_DEFAULT_3
IXC_DEFAULT_4       RRS_DELAYEDUR_1     RRS_MAINUR_1
RRS_RESTART_1       RRS_RMDATA_1        SYSARC_PLEXO_RCL
SYSIGGCAS_ECS       SYSTEM_LOGREC(NEW)  SYSTEM_OPERLOG(OLD)
SYSZWLM_WORKUNIT

-D XCF,CF,CFNM=CF3
IXC362I 15.58.03 DISPLAY XCF 995
CFNAME: CF3
STRUCTURES:
ISTGENERIC          SYSTEM_LOGREC(OLD)  SYSTEM_OPERLOG(NEW)
SYSZWLM_991E2094

```

Figure 3-12 The result of the REALLOCATE process

When the upgrade action or maintenance finishes and the CF is back in the sysplex, the CF must be placed out of the maintenance mode in order to accept allocation of structures. You might use the following command to take a CF out of maintenance mode:

```
SETXCF STOP,MAINTMODE,CFNAME=(cfname1,[cfname2,...])
```

Figure 3-13 displays the stop maintenance mode command output.

```

-SETXCF STOP,MAINTMODE,CFNAME=CF1
IXC369I THE SETXCF STOP MAINTMODE REQUEST FOR COUPLING FACILITY
CF1 WAS SUCCESSFUL.

```

Figure 3-13 Stop maintenance mode command output

The **REALLOCATE** command can be used to move structures back into the CF, as shown in Figure 3-14 on page 51, with the distribution of the allocated structures on CF1, CF2, and CF3 after the **REALLOCATE** process.

```

-D XCF,CF,CFNM=CF1
IXC362I 16.12.01 DISPLAY XCF 334
CFNAME: CF1
STRUCTURES:
DB8FU_LOCK1          DB8FU_SCA          IGWLOCK00
ISGLOCK              ISTMNPS             IXC_DEFAULT_1
IXC_DEFAULT_3        RRS_DELAYEDUR_1    RRS_MAINUR_1
RRS_RESTART_1        RRS_RMDATA_1       SYSIGGCAS_ECS
SYSTEM_LOGREC(OLD)   SYSTEM_OPERLOG(NEW) SYSZWLM_WORKUNIT

-D XCF,CF,CFNM=CF2
IXC362I 16.12.06 DISPLAY XCF 336
STRUCTURES:
EJESGDS_WTSCPLX4     IXC_DEFAULT_2       IXC_DEFAULT_4
SYSARC_PLEXO_RCL     SYSTEM_OPERLOG(OLD)

-D XCF,CF,CFNM=CF3
IXC362I 16.12.09 DISPLAY XCF 338
CFNAME: CF3
STRUCTURES:
ISTGENERIC           SYSTEM_LOGREC(NEW)  SYSZWLM_991E2094

```

Figure 3-14 CF1, CF2, and CF3 after the REALLOCATE process

## Modifications to existing messages

Existing messages for structure allocation evaluation are updated to support a new reason for not using a CF that is in maintenance mode. The following messages are updated:

IXC574I - Message IXC574I is issued for the specified structure during structure allocation of the new instance during a CF Structure rebuild or allocation process.

IXC463I - Message IXC463I is issued when the system attempted to allocate the structure, but no Coupling Facility was suitable.

IXL015I - Message IXL015I is issued when a program attempted to connect or rebuild-connect to a Coupling Facility structure and the connect processing returned structure allocation information.

IXC369I - Message IXC369I is issued with the result of the SETXCF command processing for each CF

IXC569I - Message IXC569I is issued as hard copy message only to the syslog whenever a Coupling Facility has been placed into or taken out of Maintenance Mode.

IXC361I - Message IXC361I is issued when a DISPLAY XCF,CF command was entered to display summary information about the Coupling Facilities defined in this sysplex. A new status line will be issued when the CF is in Maintenance mode.

IXC362I - Message IXC362I is issued when a DISPLAY XCF,CF command was entered to display detailed information about the Coupling Facilities defined in this sysplex. A new status line will be issued when the CF is in Maintenance mode.

Archived

## ICSF support for PKCS #11

RSA Laboratories of RSA Security Inc. offers its Public Key Cryptography Standards (PKCS) to developers of computers that use public key and related technology. PKCS #11, also known as Cryptoki, is the cryptographic token interface standard. It specifies an application programming interface (API) to devices, referred to as *tokens*, that hold cryptographic information and perform cryptographic functions. The PKCS #11 API is an industry-accepted standard commonly used by cryptographic applications.

With z/OS V1R9, ICSF supports PKCS #11, which provides an alternative to the IBM Common Cryptographic Architecture (CCA) and broadens the scope of cryptographic applications that can make use of zSeries cryptography. PKCS #11 applications developed for other platforms can be recompiled and now run on z/OS.

In this chapter, the PKCS #11 support in z/OS is introduced as follows:

- ▶ PKCS #11 overview
- ▶ z/OS ICSF overview
- ▶ ICSF: PKCS #11 support

## 4.1 PKCS #11 overview

PKCS #11 specifies an application programming interface (API) to devices (virtual or real), referred to as tokens. *Tokens* hold cryptographic information and perform cryptographic functions. PKCS #11 was designed by RSA as a standard for talking to smart cards. The major advantage of PKCS #11 over other, competing standards such as OpenSSL is that the persistent storage and retrieval of objects is part of the standard, where objects are certificates, keys, and even application-specific data objects.

On most single-user systems, a token is a smart card or other plug-installed cryptographic device, accessed through a card reader or slot. The PKCS #11 specification assigns numbers to slots, known as *slot IDs*. An application identifies the token that it wants to access by specifying the appropriate slot ID. On systems that have multiple slots, it is the application's responsibility to determine which slot to access. PKCS #11 is becoming very popular on other platforms.

z/OS must support multiple users, each potentially needing a unique keystore. In this multiuser environment, the system does not give users direct access to the cryptographic cards installed as if they were personal smart cards. Instead, z/OS PKCS11 tokens are virtual, conceptually similar to RACF (SAF) key rings. An application can have one or more z/OS PKCS11 tokens, depending on its needs.

Typically, PKCS #11 tokens are created in a factory and initialized either before they are installed, or upon their first use. In contrast, z/OS PKCS11 tokens can be created using system software such as RACF, the `gskkyman` utility, or by applications using the C API. Each token has a unique token name, or label, that is specified by the end user or application at the time that the token is created.

### PKCS #11 terminology

Following are some terms that are defined in PKCS #11.

<b>TOKEN</b>	Logical view of a crypto device: for example, a smart card
<b>SLOT</b>	Logical view of a card reader; numbered 0-n
<b>OBJECT</b>	Item stored on a token; for example a certificate or, key
<b>USER</b>	Owens the private data on the token knowing the PIN
<b>Security Officer (SO)</b>	Person who initializes a token

**Note:** Additional information on PKCS #11 can be found at the following URL:

<http://www.rsa.com/rsalabs/node.asp?id=2133>

## 4.2 z/OS ICSF overview

ICSF is a software element of z/OS. ICSF works with the hardware cryptographic features and the Security Server (RACF element) to provide secure, high-speed cryptographic services in the z/OS environment. ICSF provides the application programming interfaces by which applications request the cryptographic services. ICSF is also the means by which the secure cryptographic features are loaded with master key values, allowing the hardware features to be used by applications. The cryptographic feature is secure, high-speed hardware that performs the actual cryptographic functions. Your processor hardware determines the cryptographic feature available to your applications.

**Note:** ICSF, the cryptographic element of z/OS, only provides support for Common Cryptographic Architecture (CCA). This puts z/OS at a disadvantage when it comes to application enablement. PKCS #11 is not as low-level as CCA is, so this makes it easier to use with C- based applications. With some minor exceptions, RACF, System SSL, and Java (JSSE) all use different key stores with RACF key rings, System SSL key databases, and Java key stores. Providing PKCS #11 native to ICSF provides an opportunity to have a common key store usable by all three.

## 4.3 ICSF: PKCS #11 support

Prior to z/OS V1R9, z/OS ICSF provided the interface to the cryptographic hardware on System z servers. The application programming interface (API) used is the IBM Common Cryptographic Architecture (CCA). In order to broaden the scope of cryptographic applications that are able to make use of zSeries cryptography, ICSF on z/OS V1R9 provides support for an additional API called Public Key Cryptography Standards #11 (PKCS #11).

PKCS #11 is an API commonly used in cryptographic applications by developers of computer systems employing public key and related technology. PKCS #11 is the cryptographic token interface standard. It specifies an API to devices which hold cryptographic information and perform cryptographic functions. It supports a new VSAM data set similar to the existing ICSF cryptographic keys data set (CKDS) and public keys data set (PKDS). This new data set, the token data set (TKDS), is the repository for cryptographic keys, certificates, and data used by PKCS #11 applications. With the TKDS and with token management callable services provided by ICSF, IBM middleware products such as RACF, SSL, and Java can use a common repository for keys and tokens and deprecate usage of their individual key stores.

### 4.3.1 PKCS #11 integration into z/OS

PKCS #11 is now a subcomponent of ICSF. To integrate PKCS #11 into ICSF, the following changes needed to be made to the standard support.

- ▶ With standard PKCS #11, security is controlled by knowledge of a PIN. This does not work well on a multi-user system because multiple applications and users must share same PIN.

With PKCS #11 on z/OS, however, tokens are not cryptographic devices but rather virtual smart cards. New tokens can be created at any time. These tokens can be application-specific or system-wide, depending on the RACF access control you have defined. PINs are not used. The token access in z/OS is controlled by a new CRYPTOZ RACF resource class.

- ▶ With standard PKCS #11, applications store keys on smart cards. There is only so much space on a single cryptographic card with z/OS. z/OS applications should not have to know what cards are available, because ICSF does not provide direct access to cards.

With this new support, tokens are virtual smart cards. Each is a collection of certificates, keys, and data objects as needed by a given application such as a RACF key ring.

Token names (labels) can be up to 32 characters (such as A-Z, 0-9, the period (.), @, #, and \$). Applications do not access crypto cards directly. In ICSF, tokens and their contents are stored in a new VSAM data set, called the token key data set (TKDS). Now, ICSF has three VSAM data sets, CKDS, PKDS, and the new TKDS. In addition to the PKCS #11 C API, there are also some low level callable services.

The RACF **RACDCERT** command and System SSL's **gskkyman** utility can be used to create and manage tokens.

The token browser provides a means of examining PKCS #11 tokens that may have been created or modified outside of ICSF (for example, by an application program). The browser is not intended for full token management; instead, you can use RACF or System SSL for that. However, it does have some minor editing capability, such as allowing you to alter object labels.

**Note:** This new support is useful for PKCS #11 applications that are ported to z/OS.

### 4.3.2 Updating your ICSF definition to support PKCS #11

A new RACF (SAF) class is created for defining the token protection profiles called CRYPTOZ. There are two CRYPTOZ class resources for each token: one for the SO role, and one for the user role, as shown in Table 4-1.

- ▶ The RACF token-specific resources are for a USER role and a SO role.
- ▶ The three PKCS #11 access types for SAF access levels are User R/W, SO R/W, and User R/O.
- ▶ The three z/OS unique access types are Weak SO, Weak User, and Strong SO.

The two roles and three levels provide the six possible access levels. The User R/O, SO R/W, and User R/W access levels are the standard PKCS #11 roles. For example, the security officer (SO) R/W can initialize tokens. The Weak SO, Weak User, and Strong SO access levels are unique to z/OS.

Table 4-1 Token access control using the CRYPTOZ resource class

CRYPTOZ resource	READ	UPDATE	CONTROL
SO token-label	<b>Weak SO</b> - read/create/delete/modify/use public objects	<b>SO R/W</b> - Weak SO plus create/delete tokens	<b>Strong SO</b> - SO RW plus read (but not use) private objects, create/delete/modify private objects
User token-label	<b>User R/O</b> - read/use public and private objects	<b>Weak User</b> - User R/O plus create/delete/modify private and public objects (cannot add/delete/modify certificate authority objects)	<b>User R/W</b> - Weak User plus add/delete/modify certificate authority objects

#### Using the resource class access

The resource names are formed from the label of the token being protected: SO for the security officer role, and USER for the User role. Generic profiles can be used to protect multiple tokens. The access levels unique to z/OS are as follows.

**Weak SO** With this resource you can define the trust policy for a token (the trusted CA certificates), but cannot initialize tokens. This level of access might be appropriate for a corporate trust policy officer or auditor.



- Strong SO** With this resource you can initialize tokens and populate them, but not use the keys. This level of access might be appropriate for an application administrator.
- Weak User** With this resource you have access to everything in the token, but cannot alter the trust policy of the token. This level of access might be appropriate for a server daemon user ID.

### Implementing PKCS #11 support

You need to provide the following two actions on an existing ICSF definition to support PKCS #11:

- ▶ You have to define a TKDS VSAM data set. ICSF provides a sample TKDS allocation job (member CSFTKDS) in SYS1.SAMPLIB. The TKDS must be a key-sequenced data set with spanned variable length records and must be allocated on a permanently resident volume.
- ▶ The TKDSN option is the name of an existing TKDS or an empty VSAM data set to be used as the TKDS. If the TKDSN option is not specified in the ICSF installation options data set, no PKCS #11 services will be provided.

### PKCS #11 tokens and objects

PKCS #11 tokens and objects are stored in a VSAM data set called the token data set (TKDS). The TKDS contains individual entries for each token and object that is added to it. ICSF maintains two copies of the TKDS: a disk copy, and an in-storage copy. Only token objects are stored in the TKDS; session objects are stored in a data space.

New tokens can be created at any time. These tokens can be application-specific or system-wide, depending on the RACF access control you have defined.

## 4.3.3 RACF and z/OS PKCS #11 token services

Tokens are containers that hold digital certificates and keys. z/OS supports PKCS #11 tokens with tokens provided and managed by ICSF.

Figure 4-1 on page 58 shows an architectural view of the entire support. Everything below the dashed line is the ICSF address space. Above the dashed line is the users' address space. The human user interfaces are the token browser panels, for the **RACDCERT** command and the **gskkyman** utility.

The others are programming interfaces shown in Figure 4-1 on page 58 at the callable service level.

- ▶ The C application-level programming interfaces are distributed as follows.
  - 31-bit, 31-bit XPLINK, 64-bit DLLs and sidedecks shipped in SYS1.SIEALNKE.
  - UNIX versions also shipped in /usr/lib and /usr/lpp/pkcs11/lib.
  - csnpdefs.h is shipped in SYS1.SIEAHDR.H and /usr/include.
  - Sample code and makefiles are shipped in /usr/lpp/pkcs11/samples.
- ▶ The existing SSL applications (both System SSL and JSSE) can make use of tokens in place of key stores. System SSL's **gskkyman** is another utility for managing certificates and keys. The **gskkyman** UNIX command line utility is enhanced to manage tokens similar

to key database (.kdb) files. Like **RACDCERT**, it too has been modified to provide token management support.

While **gskkyman** has token support, the System SSL runtime services have not been modified to support reading tokens directly through SAF key ring services and can use tokens the same as key rings. To get System SSL to read a token indirectly (through **R\_Datalib**), use the key ring naming convention as follows:

```
*TOKEN*/<+fn=Arial+fs=12+fx=0+fe=1>token-name>
```

The **gskkyman** command line options work just like their key database equivalents.

- ▶ You can authorize applications to use the **R\_datalib** (IRRSDL00 or IRRSDL64) callable service to read and extract token information. This is the service SSL applications use to read RACF key rings. Note that a caller of **R\_Datalib** (System SSL and JSSE) does *not* have to change to use tokens instead of key rings. You can tell it to read a token instead of a key ring simply by prefixing the token name with **\*TOKEN\***.

For example, a keyfile directive in Webserver's **httpd.conf** file is as follows:

```
keyfile *TOKEN*/VENDOR.TOK SAF
```

**Note:** For details, see *z/OS Security Server RACF Callable Services, SA22-7691*.

- ▶ You can use resources in the **CRYPTOZ** class to control access tokens.

**Note:** See *z/OS Cryptographic Services Integrated Cryptographic Services Facility Writing PKCS #11 Applications, SA23-2231*, for additional programming considerations.

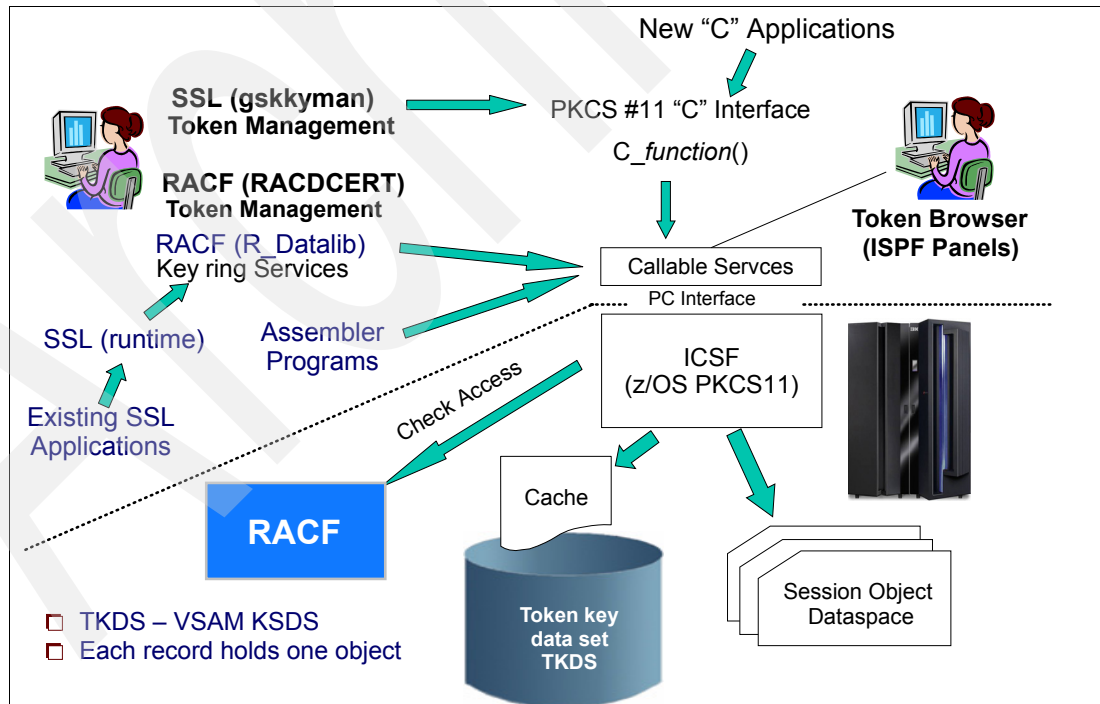


Figure 4-1 Architectural view of the entire support with ICSF and PKCS #11

## RACF RACDCERT command

The RACF **RACDCERT** command now has support for tokens, similar to the key ring support. There are six new subfunctions, all of which call the new ICSF token management callable services. You can use RACF in the following ways to define and manage certain certificate objects in a token (certificates, public keys, and private keys).

<b>ADDTOKEN</b>	Defines a new empty token.
<b>DELTOKEN</b>	Deletes an existing token and all its contents.
<b>LISTTOKEN</b>	Displays information about the objects contained in the token.
<b>BIND</b>	Connects a RACF certificate, its public key, and (in some cases) its private key, to an existing token.
<b>UNBIND</b>	Removes a certificate and its keys from an existing token.
<b>IMPORT</b>	Adds a certificate to RACF from an existing token.

ICSF will check access to TKDS functions via RACF calls for CRYPTOZ resource class. RACF will check access to certificate functions similar to other commands such as the FACILITY class checks.

See *z/OS Security Server RACF Command Language Reference, SA22-7687*, for syntax and usage information about these functions of the RACDCERT command.

## Token management

Because tokens are managed by ICSF, and not RACF, other applications can use ICSF functions to change tokens without updating the certificate information in the RACF database. Similarly, RACF changes to digital certificates already bound to a token are not reflected in the token information maintained by ICSF. Therefore, the following restrictions apply:

- ▶ Deleting, altering, or renewing a RACF certificate that is bound to a token has no effect on the equivalent token objects managed by ICSF.
- ▶ Deleting or altering a certificate object in a token has no effect on the following objects:
  - The equivalent RACF certificate
  - The equivalent certificate objects in other tokens

## Entering cryptographic objects into the TKDS

You can store public key objects, private key objects, secret key objects, certificate objects, and data objects in the token data set (TKDS) through the use of ICSF callable services. ICSF provides a set of callable services that allow applications to update the TKDS. Applications can use these services to create, delete, list, set and get attribute values from the TKDS.

The keys stored in the TKDS are not encrypted. Therefore, it is recommended that you RACF-protect data set access to the TKDS. (This is in addition to the RACF protection of the individual tokens via the CRYPTOZ class.) This will provide additional security for your installation.

Also, you can use the ICSF panel on ISPF to create, delete, manage, and list your tokens.

## 4.3.4 Migration considerations

Restarting the ICSF started procedure is required to get ICSF to recognize the setup changes. Additional changes to the RACF CRYPTOZ resource class after restarting do not require another restart.

IBM provides a sample PKCS #11 program called testpkcs11. The program is passed the name of a PKCS #11 token, and performs the following tasks:

- ▶ Creates a token that has the name passed
- ▶ Generates an RSA key-pair
- ▶ Encrypts some test data using the public part of the key-pair
- ▶ Decrypts the data using the private part of the key-pair
- ▶ Deletes the key-pair and the token

This UNIX utility is accessed via /usr/lpp/pkcs11/bin/testpkcs11. The source code is shipped in /usr/lpp/pkcs11/samples.

## 4.4 Using PKCS11 token browser utility panels

The PKCS11 token browser allows management of PKCS11 tokens and objects in the TKDS. The PKCS11 token browser is Option 5.7, shown in Figure 4-2, which on the ICSF utilities panel is Option 7:

```
PKCS11 TOKEN - Management of PKCS11 tokens
```

The user must have SAF authority to manage tokens and SAF authority to a token to manage the objects of a token.

The ICSF Token Management panel shown in Figure 4-2 has four options. The options Create a new token, Delete an existing token and Manage an existing token expect you to supply the full token name below. The option Delete an existing token will ask you to confirm the delete.

If you want to manage an existing token and are not sure of the token name, or you want to manage multiple tokens, use the List existing tokens option. This option will list all the tokens you have access to. Or, if you wish to narrow down the search, you can enter a partial token name. Any token that begins with the value specified will be listed, if you have access.

```
----- ICSF Token Management - Main Menu -----
OPTION ==> -
Enter the number of the desired option.
1 Create a new token
2 Delete an existing token
3 Manage an existing token
4 List existing tokens
Full or partial token name: _____
```

Figure 4-2 ICSF Token Management panel for PKCS #11

Two of the ICSF existing panels have changed: the ADMINCTL panel, and the OPTSTAT.OPTIONS. These panels now display the active TKDS name if it is set up.

The ADMINCTL panel, Option 4 from the primary panel, is shown in Figure 4-3 on page 61.

```

----- ICSF - Administrative Control Functions -- Row 1 to 4 of 4
COMMAND ==> _ SCROLL ==> PAGE

Active CKDS: SYS1.SC60.SCSFCKDS
Active PKDS: SYS1.SC60.SCSFPKDS
Active TKDS: SYS1.SC60.SCSFTKDS

To change the status of a control, enter the appropriate character
(E - ENABLE, D - DISABLE) and press ENTER.

FUNCTION STATUS
-----
. Dynamic CKDS Access ENABLED
. PKA Callable Services ENABLED
. PKDS Read Access ENABLED
. PKDS Write, Create, and Delete Access ENABLED
***** Bottom of data *****

```

Figure 4-3 ICSF Administrative Control Functions panel updated for TKDS

The OPTSTAT.OPTIONS, Option 3.1 from the primary panel, is shown in Figure 4-4.

```

----- ICSF - Installation Option Display -- Row 1 to 14 of 14
COMMAND ==> _ SCROLL ==> PAGE

Active CKDS: SYS1.SC60.SCSFCKDS
Active PKDS: SYS1.SC60.SCSFPKDS
Active TKDS: SYS1.SC60.SCSFTKDS

OPTION CURRENT VALUE
-----
CHECKAUTH RACF check authorized callers YES
COMPAT Allow CUSP/PCF compatibility NO
DOMAIN Current domain index or usage domain index 8
KEYAUTH Key Authentication in effect YES
CKTAUTH CKT Authentication in effect NO
SSM Allow Special Secure Mode YES
TRACEENTRY Number of trace entries active 1000
USERPARM User specified parameter data USERPARM
REASONCODES Source of callable services reason codes ICSF
PKDSCACHE PKDS Cache size in records 64
SYSPLEXCKDS Sysplex consistency for CKDS updates NO,FAIL(NO)
SYSPLEXTKDS Sysplex consistency for TKDS updates NO,FAIL(NO)
WAITLIST Source of CICS Wait List if CICS installed default
***** Bottom of data *****

```

Figure 4-4 ICSF Installation Option Display panel updated for TKDS

#### 4.4.1 Running ICSF in a sysplex environment

ICSF is supported in a sysplex environment. The CKDS, PKDS, and TKDS can be shared across systems in a sysplex.

##### TKDS management in a sysplex

The systems sharing a TKDS may be different LPARs on the same system, or different systems across multiple System z processors. It is not required to share the TKDS across a sysplex, but it is recommended to allow your application to be dynamically routed to any image safely. Each system may have its own TKDS.

A sysplex may have a combination of systems that share a TKDS, and individual systems with separate TKDSs. There is no requirement that the DOMAINS must be the same to share a TKDS.

When sharing the TKDS, observe these precautions:

- ▶ Dynamic TKDS services update the DASD copy of the TKDS and the in-storage copy on the system where it runs. The SYSPLEXTKDS option in the ICSF installation options data set provides for sysplex-wide consistent updates of the DASD copy of the TKDS, and the in-storage copies of the TKDS on all members of the sysplex sharing the same TKDS.

All members of the sysplex sharing the TKDS must be running ICSF HCR7740 or later in order to participate in the sysplex-wide consistency of TKDS data.

If SYSPLEXTKDS(YES,FAIL(xxx)) is coded in the installation options data set, a sysplex broadcast message will be issued informing sysplex members of the TKDS update, and requesting them to update their in-storage TKDS copy.

If SYSPLEXTKDS(NO,FAIL(xxx)) is coded in the installation options data set, there is no sysplex broadcast of the update.

- ▶ If multiple sysplexes share a TKDS, or if a sysplex and other non-sysplex systems share a TKDS, there is no provision for automatic update of the in-storage copies of the TKDS on the systems which are not in the same sysplex as the system initiating the TKDS update.

### Changing parameters in the installation options data set

The ICSF installation options, SYSPLEXTKDS(YES or NO,FAIL(fail-option)), can be defined and changed as follows:

<b>SYSPLEXTKDS(NO,FAIL(fail-option))</b>	Indicates no XCF signalling will be performed when an update to a TKDS record occurs.
<b>SYSPLEXTKDS(YES,FAIL(fail-option))</b>	Indicates the system will be notified of updates made to the TKDS by other members of the sysplex who have also specified SYSPLEXTKDS(YES,FAIL(fail-option)).
<b>SYSPLEXTKDS(YES,FAIL(YES))</b>	Indicates ICSF will terminate abnormally if there is a failure creating the TKDS latch set.
<b>SYSPLEXTKDS(YES,FAIL(NO))</b>	Indicates ICSF initialization processing will continue even if the request to create a TKDS latch set fails with an environment failure.  This system will not be notified of updates to the TKDS by other members of the ICSF sysplex group.

If you do not specify the SYSPLEXTKDS option, the default value is:

SYSPLEXTKDS(NO,FAIL(NO))



## Allocation dynamic storage improvements

The allocation dynamic storage enhancements are designed to improve scalability by moving dynamic area storage above the 16 Mb line. This gives allocation more space for recursive retry logic. It is done as part of allocation recovery, and reduces the chances of allocation failures and possible ABEND 878 in the allocation code. This improvement frees storage below the 16 Mb line for use by programs that do require storage with that attribute.

Also, by having more working space, improved performance of allocation can occur over time.

This chapter describes the component changes that relieve dynamic storage in all ASIDs.

- ▶ Allocation overview
- ▶ Allocation improvements in z/OS V1R9

## 5.1 Overview

*Allocation* is the process by which the system assigns, or allocates, I/O resources to your job. An I/O resource is a ddname-data set combination, with any associated volumes and devices.

*Deallocation* is the process by which the system releases, or deallocates, I/O resources that were allocated to your job.

There are two basic types of allocation: job step allocation and dynamic allocation. The two types allocate resources at different points in program processing. Job step allocation assigns resources to your program before your program runs, and dynamic allocation assigns resources to your program while it is running. The needs of your program determine which type of allocation you should use.

### **Characteristics of job step allocation**

When using job step allocation, you request I/O resources through JCL. The system allocates those I/O resources before your program runs, as part of initiating the job step, and deallocates resources after your program runs, as part of job step termination. This type of allocation ensures that the resources you request are available before your program runs, and throughout program execution.

### **Characteristics of dynamic allocation**

When using dynamic allocation, you request I/O resources by coding the DYNALLOC macro and filling in the fields of the SVC 99 parameter list. The system allocates and deallocates those I/O resources while your program is running. Dynamic allocation also allows you to request information about your allocation environment, and to deallocate or modify characteristics of your allocation environment that were acquired either dynamically or through JCL.

Dynamic allocation allows you to tailor your device allocations based on input to your program. You can design your program to dynamically allocate only those devices that are necessary in a particular programming path, rather than allocating all possible device requirements before your program runs.

Dynamic allocation also allows you to use common resources more efficiently. When there is high contention for a resource, dynamic allocation allows you to acquire an I/O resource just before you need it and to release it just after you need it, so that your program holds the resource for a shorter length of time.

### **Allocation virtual storage constraints**

In order to process any data set, the system must allocate it to your program before you can open it. There are many control blocks that allocation creates in the user private area. To handle more than 100 K devices as some DB2 users do, an excessive amount of virtual storage is used by allocation.

A great deal of effort has been invested in moving these areas above the line in previous releases, but until now some areas were still gotten in 24-bit addressing, thereby causing constraints in this area and limiting the number of data sets that could be allocated.

In the following sections we describe the allocations improvements provided by z/OS V1R9.



## 5.2 Allocation improvements in z/OS V1R9

In z/OS V1R9, all the requirements to get storage in 24-bit were eliminated, and then all these dynamic areas were moved to 31-bit addressing. The following improvements are introduced:

- ▶ Reduce the chance of S878 in storage-constrained environments. In the past some abend878 or abend80A ABENDs occurred because there was not enough available storage in 24-bit of the user private area. With this improvement the abends caused by allocation excessive virtual storage usage were eliminated, so installations will see fewer of these types of ABENDs.
- ▶ Extends limits of our recursive allocation retry processing.
- ▶ Results in fewer failed allocations causing jobs to fail. Now, some jobs that had storage constraints in 24-bit mode can allocate more data sets.
- ▶ Using storage above 16 MB frees storage for required users.
- ▶ Having more space allows the compiler to generate more efficient code.
- ▶ The storage manager was changed to obtain these areas in large buffer pools, so allocation handles large buffers more efficiently and does not need to run this process so often.

### Allocation use of storage

In past releases of MVS, the allocation component used storage below the 16 MB line for its dynamic area. The services invoked by allocation required storage with that attribute. However, over time, these requirements have been eliminated, leaving allocation still using those resources for no good reason. Even though most components allow above-the-line storage, there are still reasons to use below-the-line storage. Because installations are consolidating work on z/OS, there is less below-the-line storage free for everyone to use. Also, storage use by allocation modules is generally growing over time. And finally, by using that constrained resource, the recursive processing that helps ensure jobs get the resources they need to run is limited, resulting in abends and failed jobs.

With z/OS V1R9, allocation moves its dynamic areas where there is more room to grow above the 16 MB line. In the process of doing this, the storage manager is updated to obtain storage in larger chunks, to improve the efficiency of the load modules. The benefit of these changes is to free storage for others to use, allow more working room, and reduce the chances of 878 abends due to out-of-storage conditions.

This changes, for this line item, an intrinsic part of the allocation component. Therefore, the support is invoked by all MVS allocation invokers, namely MVS batch processing and dynamic allocation, used to allocate devices and data sets at run time.

### Allocation improvement considerations

The two types of allocation are invoked by the MVS scheduler component: one responsible for batch allocation, and the other by any other subsystem or application wanting to perform a dynamic allocation. Therefore, all changes are internal to the component, so there are no new or changed externals.

For any exploiters doing an allocation, such as the DB2 subsystem, TSO users, or batch jobs, the use of allocation exits has always been required to be AMODE 31. Those exits may now see changes in addressing due to the above-the-line storage for parameter lists.

Note that these changes only affect z/OS V1R9 systems; lower-level systems will see no changes in allocation.

Archived



## System Logger enhancements

System Logger is an MVS component that allows an application to log data from a sysplex. You can log data from one system or from multiple systems across the sysplex. A system logger application can write log data into a log stream, which is simply a collection of data.

This chapter describes a general overview and some improvements in the system logger function (IXGLOGR) in z/OS V1R9. The following topics are discussed:

- ▶ System Logger overview
- ▶ Recall processing for log stream migrated data set
- ▶ Cleanup of CF list entries for unconnected log streams
- ▶ General documentation improvements

## 6.1 System Logger overview

System logger is a set of services that allows an application to write, browse, and delete log data. You can use system logger services to merge data from multiple instances of an application, including merging data from different systems across a sysplex.

Suppose you are concurrently running multiple instances of an application in a sysplex, and each application instance can update a common database. It is important for your installation to maintain a common log of all updates to the database from across the sysplex, so that if the database should be damaged, it can be restored from the backup copy. You can merge the log data from applications across the sysplex into a log stream, which is simply a collection of data in log blocks residing in the Coupling Facility and on DASD.

### Log stream

A *log stream* is an application-specific collection of data that is used as a log. The data is written to and read from the log stream buffers by one or more instances of the application associated with the log stream. A log stream can be used for such purposes as a transaction log, a log for recreating databases, a recovery log, or other logs needed by applications.

A system logger application can write log data into a log stream, which is simply a collection of data. Data in a log stream spans two kinds of storage:

- ▶ Interim storage, where data can be accessed quickly without incurring DASD I/O
- ▶ DASD log data set storage, where data is hardened for longer term access

When the interim storage medium for a log stream reaches a user-defined threshold, the log data is offloaded to DASD log data sets. There are two types of log streams:

- ▶ Coupling Facility log streams
- ▶ DASD-only log streams

The main difference between the two types of log streams is the storage medium that system logger uses to hold interim log data:

- ▶ In a Coupling Facility log stream, interim storage for log data is in Coupling Facility list structures.
- ▶ In a DASD-only log stream, interim storage for log data is contained in local storage buffers on the system. Local storage buffers are data space areas associated with the system logger address space, IXGLOGR.

Your installation can use just Coupling Facility log streams, just DASD-only log streams, or a combination of both types of log streams. The requirements and preparation steps for the two types of log streams are somewhat different.

Some key considerations for choosing either Coupling Facility log streams or DASD-only log streams are:

- ▶ The location and concurrent activity of writers and readers to a log stream's log data
- ▶ The volume of log data written to a log stream.

Coupling Facility log streams are required when:

- There needs to be more than one concurrent log writer and/or log reader to the log stream from more than one system in the sysplex.
- There are high volumes of log data being written to the log stream.

DASD-only log streams can be used when:

- ▶ There is no need to have more than one concurrent log writer and/or log reader to the log stream from more than one system in the sysplex.
- ▶ There are low volumes of log data being written to the log stream.

**Note:** Because DASD-only log streams always use staging data sets, high volume writers of log data may be throttled back by the I/O required to record each record sequentially to the log stream's staging data sets.

### Coupling Facility log stream

Figure 6-1 shows how a Coupling Facility log stream spans two levels of storage; the Coupling Facility for interim storage, and DASD log data sets for more permanent storage. When the Coupling Facility space for the log stream fills, the data is offloaded to DASD log data sets. A Coupling Facility log stream can contain data from multiple systems, thus allowing a system logger application to merge data from systems across the sysplex.

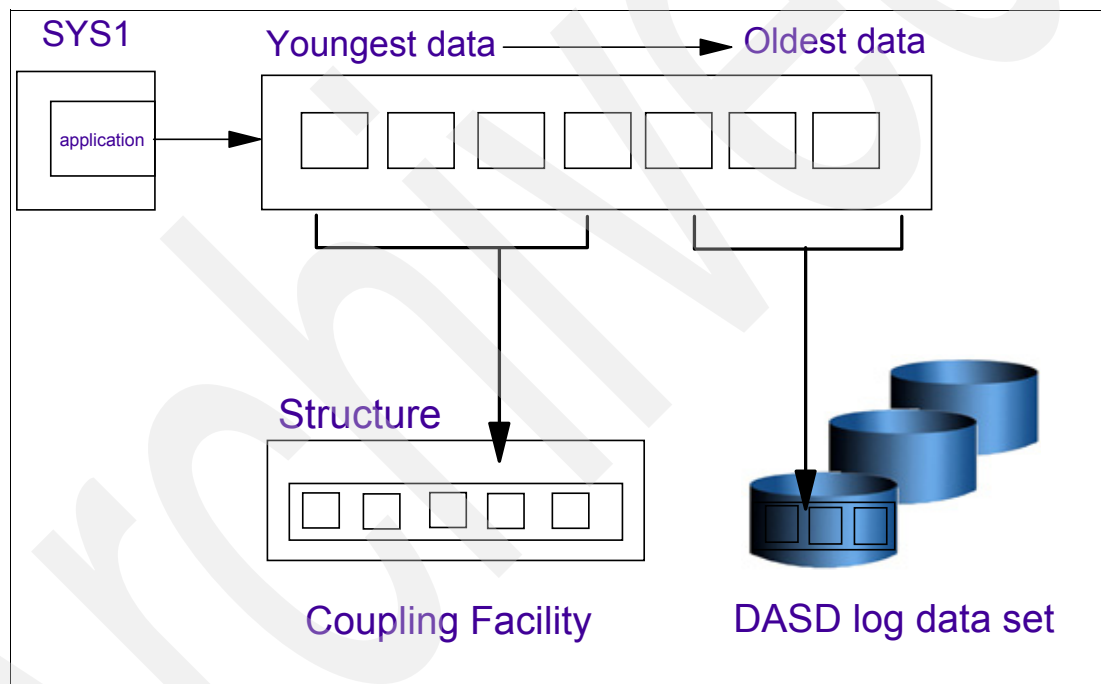


Figure 6-1 Log stream data on the Coupling Facility and DASD

When a system logger application writes a log block to a Coupling Facility log stream, system logger writes it first to a Coupling Facility list structure. System logger requires that a Coupling Facility list structure be associated with each log stream.

When the Coupling Facility structure space allocated for the log stream reaches the installation-defined highoffload threshold, system logger moves (offloads) the log blocks from the Coupling Facility structure to VSAM linear DASD data sets, so that the Coupling Facility space for the log stream can be used to hold new log blocks. From a user's point of view, the actual location of the log data in the log stream is transparent.

## DASD-only log stream

Figure 6-2 shows a DASD-only log stream spanning two levels of storage; local storage buffers for interim storage, which is then offloaded to DASD log data sets for more permanent storage.

A DASD-only log stream has a single-system scope; only one system at a time can connect to a DASD-only log stream. Multiple applications from the same system can, however, simultaneously connect to a DASD-only log stream.

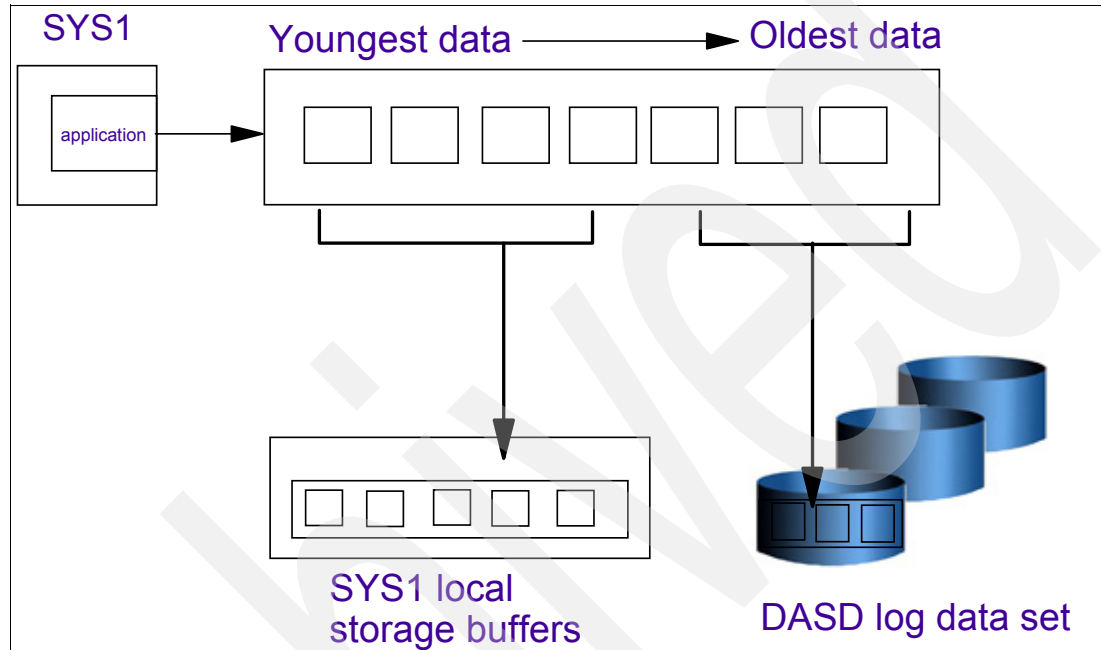


Figure 6-2 Log stream data in local storage buffers and DASD log data sets

When a system logger application writes a log block to a DASD-only log stream, system logger writes it first to the local storage buffers for the system and duplexes it to a DASD staging data set associated with the log stream. When the staging data set space allocated for the log stream reaches the installation-defined highoffload threshold, system logger offloads the log blocks from local storage buffers to VSAM linear DASD data sets. From a user's point of view, the actual location of the log data in the log stream is transparent.

Both a DASD-only log stream and a Coupling Facility log stream can have data in multiple DASD log data sets; as a log stream fills log data sets on DASD, system logger automatically allocates new ones for the log stream.

### 6.1.1 Log stream exploiters

There are several log stream exploiters, each one with very specific use; there is no unique recommendation to define a log stream. You must follow the application's recommendations to define each log stream.

CICS uses two log streams (DFHLOG and DFHSHUNT) for transaction recovery. As soon the transaction ends, it deletes all entries from log stream. It is acceptable that all these log stream entries reside in the interim buffer and do not go to the DASD log data set; if you handle all the CICS log stream entries in the buffer, you will enjoy better performance.

OPERLOG uses a log stream to record all the SYSLOG messages for long periods of time on data sets. These data sets usually are managed by HSM or a similar product.

Any application using IXGLOGR can retrieve data from the log data that resides on DASD; when they are migrated, it calls the HSM to recall them.

Prior to z/OS V1R7, there was just one task in the IXGLOGR ASID for recalls that were performed synchronously. If several recall requests arrived at the same time, a recall queue was built up. If a problem occurred with the first recall on the queue, then the remainder of the queue would wait for a long period of time until the problem was corrected, thus causing issues for applications. Also, there was a performance problem if several requests came at the same time.

### 6.1.2 z/OS V1R8 improvements of log stream data sets recall

Starting with z/OS V1R8, you can separate log streams into two groups, PRODUCTION and TEST. This means that you can separate system logger processing for TEST log streams from your PRODUCTION work log streams on a single system or sysplex. Your PRODUCTION log streams are then protected from a hang or failure in the System Logger test environment.

To separate log streams into PRODUCTION and TEST log streams, you can use the GROUP(PRODUCTION | TEST) parameter in the following ways:

- ▶ To group log streams using a batch program, use the GROUP(PRODUCTION | TEST) parameter on the DEFINE or UPDATE requests on the administrative data utility, IXCMIAPU.
- ▶ To group log streams using a program, use the new GROUP(PRODUCTION | TEST) parameter on the DEFINE or UPDATE requests on the IXGINVNT service.

When you have log streams separated into PRODUCTION and TEST groups, System Logger will do data set processing, such as data set allocations, data set recalls, and other functions for the two log stream groups in two different sets of tasks. In addition, system logger limits resource consumption of TEST log streams as follows:

- ▶ TEST log streams are limited to using a maximum of 25% of the connections allowed, whereas PRODUCTION log streams can use at least 75% of connection slots.
- ▶ TEST log streams are limited to using a maximum of 25% of LOGR couple data set extents allowed, whereas PRODUCTION log streams can use at least 75%.

By default, log streams are PRODUCTION log streams. This means that existing log streams with no GROUP designation are PRODUCTION log streams.

#### Using structures with grouped log streams

A Coupling Facility structure can only have one type of log stream assigned to it, either TEST or PRODUCTION. If you try to assign a TEST log stream, for example, to a STRUCTURE with PRODUCTION log streams, the request will fail with a reason code of IxgRsnCodeBadGroup (X'08E9').

The first log stream that is defined to a structure determines what type of log streams can be defined to that structure. If the first log stream defined to a structure is a TEST log stream, you can only define TEST log streams to that structure. If you specify or default to PRODUCTION for the first log stream defined to a structure, you can only define other PRODUCTION log streams to that structure.

## Other support for separating TEST and PRODUCTION log streams

The following interfaces support the separation of TEST and PRODUCTION log streams:

- ▶ The **DISPLAY LOGGER** command displays the group designation for log streams.
- ▶ The IXCMIAPU utility LIST LOGSTREAM request displays the group designation for log streams.
- ▶ SMF record type 88, subtype 1, Log Stream section, includes field SMF88GRP to display the group designation for each log stream.
- ▶ IXGQUERY will return the log stream group designation as long as you specify a large enough buffer length (200 bytes or greater).
- ▶ ENF signal 48 for DEFINE and UPDATE log stream requests will identify the group of the log stream.

## z/OS V1R8 log stream recall processing limitations

Although z/OS V1R8 increased the log stream recall capacity using two groups, it did not solve the performance or hung problems of the log stream recall processing because the same situation could occur in each group.

System Logger has single-threaded, synchronous handling of recall requests for migrated log stream data sets. This means that each data set recall must be satisfied (successfully or otherwise) before the next migrated data set recall is requested. This occurs in two tasks; one for the PRODUCTION group, and one for the TEST group.

This can cause limited or slower access to the log stream resources. For example, a recall request could be necessary during a log stream's offload activity, or when an application is browsing log data that can result in interference between different log stream activities. A recall request for one log stream data set could hold up a recall request for another log stream data set. These problems are addressed in z/OS V1R9, as described in the following section.

## 6.2 z/OS V1R9 improvements of log stream data set recalls

z/OS V1R9 continues using two sets of tasks to process the PRODUCTION and TEST log streams. Now, however, the recalls are processed asynchronously and up to 24 concurrent requests for PRODUCTION log streams are allowed, and up to 8 concurrent requests for TEST log streams are allowed.

DFSMSHsm or an equivalent function is required. The ability to display data sets being recalled by System Logger and to have System Logger stop waiting on a data set recall are enhanced. This provides relief for all System Logger exploiters when an installation makes use of data set migration/recall capabilities, and it helps to reduce the interference previously caused by the recall request of one log stream data set needing to be completely satisfied before System Logger starts the next recall request.

### New command support

New command parameters are created to manage the recall environment:

- ▶ Use the **D LOGGER, ST, REC** command to see the status of the recalls as shown in Figure 6-3 on page 73.  
**RECALLS** or **REC** - This new keyword is a filter that requests a display of all the outstanding asynchronous recall requests that system logger has made to DFSMSHsm using the ARCHRCAL service.



This command also shows data set recalls waiting for a significant number of seconds.

```
D  LOGGER,ST,REC
IXG601I  16.52.38  LOGGER DISPLAY 719
SYSTEM LOGGER STATUS
SYSTEM  SYSTEM LOGGER STATUS
-----
SC70    ACTIVE
        LOGGER DATA SET RECALLS
        GROUP: PRODUCTION
        SECONDS DATA SET NAMES
        00000038 IXGLOGR.PROD.STREAM01.A0000001
        00000137 IXGLOGR.PROD.STREAM35.A0000041
        GROUP: TEST
        NO DATA SET RECALLS WAITING
```

Figure 6-3 *Display logger shows the recalls pending progress*

- Use the **SETLOGR FORCE** command to clean up log stream or data set resources related to a system logger log stream. The command is useful for managing a log stream when a log stream becomes unusable. The command is also useful for causing Logger to no longer wait on a particular migrated data set being recalled. System Logger will attempt to release all the related resources for the log stream or data set based on the request.

Use the **SETLOGR FORCE** command to stop waiting on an outstanding asynchronous recall request for a named data set, as shown in Figure 6-4. The new parameter **NOREC** or **NORECALL** directs Logger to stop waiting on an outstanding asynchronous recall request for the named data set and displays if the recall has been waiting too long. It can be forced, so applications waiting on the recall request can continue.

Recall requests can also hold up offloads, and messages IXG310, IXG311, and IXG312 can be shown on the console. In these situations, a **SETLOGR FORCE, NORECALL** request can be issued to stop waiting on the recall, to allow the affected applications to continue processing.

```
setlogr force,norec,dsn=IXGLOGR.PROD.STREAM01.A0000025
IXG651I SETLOGR FORCE NORECALL COMMAND ACCEPTED FOR
DSNAME=IXGLOGR.PROD.STREAM01.A0000025
IXG280I IXGLOGR RECALL REQUEST STOPPED BY SETLOGR COMMAND FOR
DSN=IXGLOGR.PROD.STREAM01.A0000025
IXG661I SETLOGR FORCE NORECALL PROCESSED SUCCESSFULLY FOR
DSNAME=IXGLOGR.PROD.STREAM01.A0000025
```

Figure 6-4 *SETLOGR FORCE recall command example*

Use the **SETLOGR FORCE** command when a System Logger service task is not progressing properly and you receive some of the following messages:

IXG281I, IXG272E, IXG312E and IXG115A

When more than one request is waiting on the same outstanding recall, the **SETLOGR FORCE** command will affect all them. Note that these two commands only execute in z/OS V1R9; in previous releases, a syntax error occurs. This implementation is completely transparent to the System Logger exploiters, they will get this benefit without any change.

## 6.3 Cleanup of CF list entries for unconnected log streams

In previous releases, if the LOGR CDS indicated a log stream was never connected or there are no current systems connected (and no failed persistent connections), then there should be no CF structure list header information or entries and elements on the lists associated with the log stream. If this mismatch occurred, the application could not connect to the log stream structure and it would fail. A similar situation could occur with the log stream structure if a failed persistent connection occurred and it did not match with the LOGR CDS view.

In both cases, there was no way to clean up log stream structure header information so as to allow the application to connect to the structure. The System Logger should ensure the list headers for a log stream are established as in a new state when a log stream had no current connections.

With z/OS V1R9, if there is a mismatch of information between the LOGR CDS view and the log stream structure header information, the first system that tries to connect to a log stream structure will use the LOGR CDS view information to clean up the log stream structure header information, thus allowing the application to connect to structure. If the cleanup is successful, then the connection to the log stream can continue. If the cleanup is not successful, then System Logger fails the log stream connection with an error return and reason code information along with diagnostic messages to the hardcopy log to aid the invoking program and installation.

This implementation helps to improve System Logger reliability and availability.

## 6.4 System Logger publication updates

In z/OS V1R9, informational updates are made to publications that have System Logger descriptions:

- ▶ Using the LOGR CDS has more explicit guidance on sysplex IPL implications.
- ▶ An authorized services guide section was created and moved from an unauthorized publication to *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608. The new chapter is “Using System Logger Services”. Also added is ENF48 usage descriptions.
- ▶ Section “A.2.4.2 LOGR couple data set use considerations” was rewritten in *z/OS MVS Setting up a Sysplex*, SA22-7625.
- ▶ The explanation of message IXC287I now refers to a rewritten section “LOGR couple data set use considerations” in *z/OS MVS System Messages, Volume 10 (IXC-IZP)*, SA22-7625.
- ▶ The chapter “Using System Logger Services” was updated and this relevant section was moved to *z/OS MVS Programming: Assembler Services Guide*, SA22-7627. Another topic, “Expired Log Stream Token”, was added.
- ▶ Other documentation changes for the z/OS V1R9 enhancements are in the following publications:
  - *z/OS MVS System Commands*, SA22-7627
  - *z/OS MVS Programming: Assembler Services Reference, Volume 2 (IAARR2V-XCRLX)*, SA22-7607
  - *z/OS MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDTINFO-IXGWRITE)*, SA22-7610

## SMF recording to log streams

System Management Facility (SMF) collects and records system and job-related information for an installation. SMF formats the information that it gathers into system-related records (or job-related records). System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information on the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session.

To record SMF records in SMF data sets, an installation must allocate direct access space and catalog the SMF data sets. IBM recommends that you catalog the SMF data sets in the master catalog. SMF should have a minimum of two data sets for its use, and IBM recommends that you run with a minimum of three SMF data sets to ensure availability.

With z/OS V1R9, SMF can optionally utilize the System Logger to record SMF records into log streams, which can improve the write rate and increase the volume of data that can be recorded.

In this chapter we introduce the new SMF records write capability to a log stream using the System Logger facility.

- ▶ SMF overview
- ▶ SMF exploring log streams
- ▶ Processing SMF log streams

## 7.1 SMF overview

SMF is a system component that captures system execution informations and records them into the SYS1.MANx data set. Also, it has one interface that any application can invoke and report their performance flow and delays. This information is captured in a user-defined interval time and recorded in a data set for future processing.

Some installations use the SMF information simply to manage their system and make performance adjustments in their system definitions. Others installations use the SMF information to bill their customers.

The amount of SMF record data has increased over the time. SMF writes these records in one VSAM data set called the SYS1.MANx data set, shown in Figure 7-1, and when it fills up it switches to another one and marks the first one as dump required.

When a subsystem or user program wishes to write an SMF record, they invoke the SMF record macro, SMFEWTM. This macro takes the user record and invokes SMF code to locate an appropriate buffer in the SMF address space and copy the data there, as shown in Figure 7-1. If the record is full, another SMF program is scheduled to locate full SMF buffers and write them to the SYS1.MANx data set. Each buffer is numbered to correspond to a particular record in the SMF data set. This allows the records to be written in any order and to place them correctly in the data set.

Although this is not shown in Figure 7-1, when all records have been written and the SYS1.MANx data set is full, SMF switches to a new SYS1.MANx data set, and marks the full one as DUMP REQUIRED. That data set cannot be used again until it is dumped and cleared. Scheduling the SMF dump program must be done in a timely manner to ensure that the SMF MANx data set is returned to use as soon as possible to ensure that no data is lost due to an “all data sets full” condition.

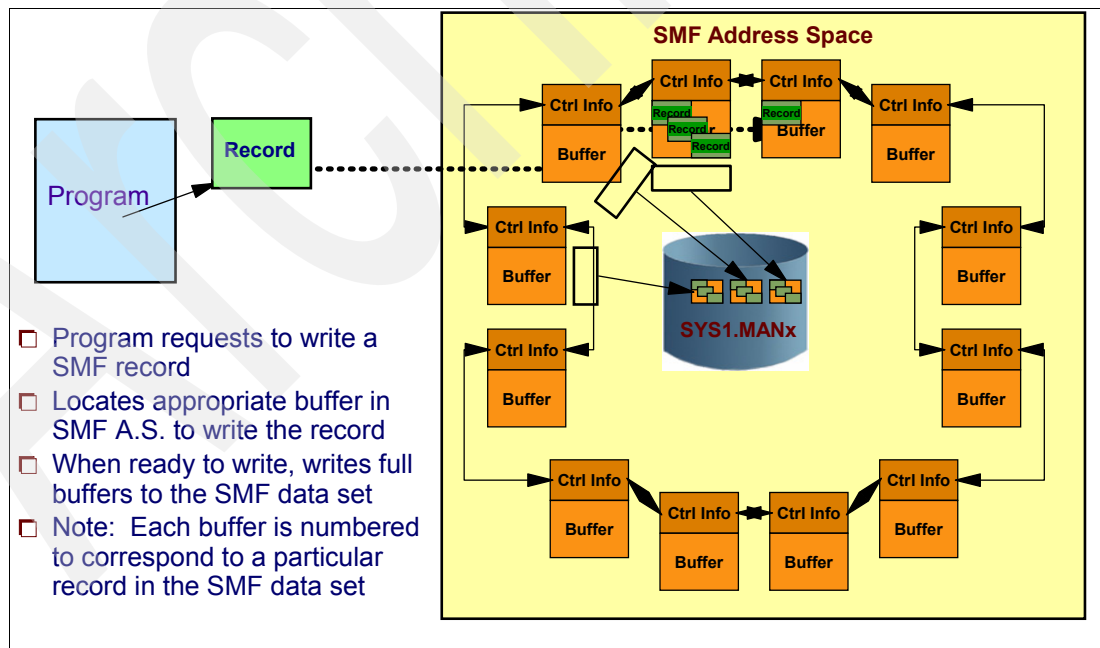


Figure 7-1 Overview of current SMF data record flow

## Current implementation deficiencies

The current implementation deficiencies are addressed by the new support in z/OS V1R9. Following are the current problems that have new options:

- ▶ Several SYS1.MANx data sets can be defined. SMF records may be lost if all the SYS1.MANx data sets are filled up and not dumped, or during the SMF switch data set.
- ▶ There are situations when that system can generate a significant amount of SMF records, exceeding the system capacity to write them into the SYS1.MANx data sets.
- ▶ The current implementation of SMF can lose data if writes are being held up, when all SYS1.MANx data sets have become full or across SMF data set switch processing.
- ▶ In addition to the possibility of losing data when recording, the SMF dump program is required to read and write every record to move the data to archive data sets, where it can then be processed by other programs (such as for sorting), or by the SMF dump program again to further filter and partition the data for use. This can result in the data being read by the SMF dump program several times, as it is read and copied for use by the various exploiters of SMF data.
- ▶ The SMF records are written in any order, so they need to be sorted for post processing and in a sysplex environment, it is necessary to merge all system information in order to have a sysplex-wide analysis.
- ▶ Many installations have already set up automation to dump the SMF data sets using IFASMFDP as a post processor and they are satisfied with this function. This support continues the same. However, for installations that need more SMF data record write capacity, or for those whose SMF records are lost during a data set switch, it is unacceptable.

### 7.1.1 SMF and log streams with z/OS V1R9

z/OS V1R9 introduces an additional capability to write SMF records to log streams managed by System Logger. With this new capability you can define several log stream for several groups of SMF records. You can define one log stream to write just the RMF records types, so during the post processor they are already isolated. When you define one log stream you have to define the SMF records type that will be written to this log stream. You can define a default log stream to write all the remaining SMF records types not defined to a specific log stream.

This new support uses the following functions and improvements:

- ▶ Utilize System Logger to improve the write rate and increase the volume of data that can be recorded.
- ▶ System Logger utilizes modern technology such as the Coupling Facility and media manager to write more data at much higher rates than the current SMF SYS1.MANx data set allows.
- ▶ Provide better management of the data by enhancing SMF to record its data to multiple System Logger log streams, based on record type. The record data is buffered in dataspaces, instead of the SMF address space private storage, thus allowing increased buffering capacity.
- ▶ Providing keywords on the OUTDD keyword of dump program that allows data to be read once and written many times. By recording different records to different log streams, SMF dump processing is also improved because a dump program per log stream can be submitted, with each being more efficient, since less records are read and ignored. This reduces processing time because there is less filtering needed by the dump program.

- ▶ One SMF record type can be written to more than one log stream. By selecting log streams based on record type, the data can be partitioned at the point of its creation, resulting in less reprocessing by the SMF dump program, which means less data read per dump program instance.
- ▶ For each SMF log stream, a dataspace is created as a buffer in the SMF ASID, so each buffer is 2 GB. Within SMF, each dataspace will have a task dedicated to writing its data to a particular log stream, increasing the rate at which you can record data to the Logger.

### z/OS V1R9 SMF recording to log streams

When recording to log streams, as shown in Figure 7-2, subsystems or user programs still invoke the SMFEWMT macro to take the user record and invoke SMF code. However, instead of locating a buffer in SMF private storage, SMF locates a dataspace corresponding to the user's record type and log stream where the record will be written. A buffer with space to hold the record is located and the record is copied there. When the record is full, a writer task is posted.

Unlike the scheduled approach in SMF data set recording, this task is already started and ready to write. In addition, writes to System Logger are done at memory-to-memory speeds, with System Logger accumulating many, many records to write out, resulting in an improved access currently not possible with current SMF data set recording.

Using a dataspace to hold the records for a given log stream allows a full 2 GB of pageable memory to be used for buffering records in the event of delays in log stream writing in System Logger. This allows more data to be buffered than SMF data set recording, which is limited to the amount of available private storage in the SMF address space.

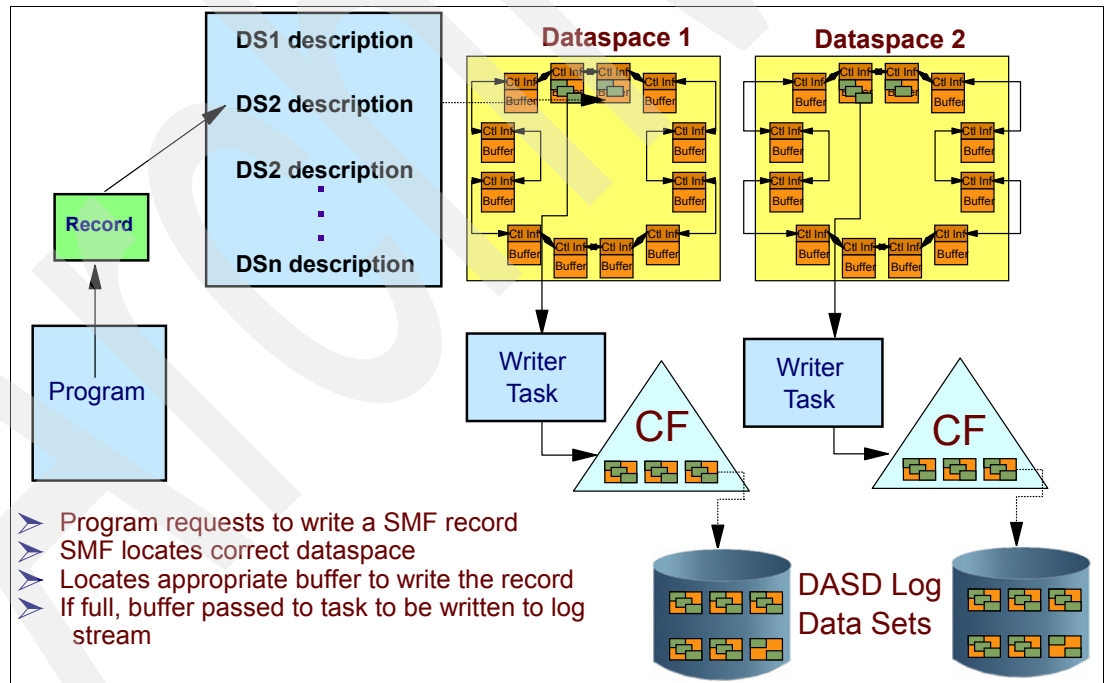


Figure 7-2 SMF data flow using log streams

The benefit in all this is that you can write more data faster, with more functionality. The System Logger was created to handle large volumes of data. With minimal SMF switch processing and no record numbering schemes to maintain, this eliminates the switch SMF command bottleneck.

**Note:** The use of log streams for SMF Data is optional. Existing SYS1.MANx function continues to exist for installations satisfied with this functionality.

## Coupling Facility or DASD-only log streams

There are two types of log streams, and SMF logging supports both of them:

- ▶ Coupling Facility log streams  
Data is stored in a Coupling Facility structure and then offloaded to DASD. A Coupling Facility log stream is ideal for merging SMF data from multiple systems. Make sure that a system's SMF ID (SID) is unique within the sysplex.
- ▶ DASD-only log streams  
Data is stored in local storage buffers and then offloaded to DASD. DASD-only log streams can only be single-system in scope, and only one system can write data to any given DASD-only log stream.

## 7.2 Installation of SMF log streams

The following items are prerequisites for installation of this new function to use SMF recording of log data using the System Logger and log streams.

### IXCMIAPU utility

Use the IXCMIAPU utility with TYPE=LOGR to create the log streams. Remember to plan the retention period, offload data set size, staging data set size, and whether the log stream will be CF-based or DASD-only.

Ensure sufficient SMS resources for peak recording periods and offload data sets.

### SMFPRMxx parmlib member

Update the SMFPRMxx parmlib member to define the use of log streams and consider the following criteria:

- ▶ Plan to retain your SMF MANx data sets as a fallback plan.
- ▶ Update the procedures for SMF SWITCH processing and IEFU29 and IEFU29L exits.
- ▶ Update the procedures to indicate how the data will be archived when using log streams.

### 7.2.1 Defining SMF log streams

Define the log streams and Coupling Facility structures (for Coupling Facility log streams) in the LOGR policy couple data set using the administrative data utility (IXCMIAPU) utility. You need to define one new log stream in the LOGR couple data set (CDS) for each LSNAME statement defined and another (optional) for the DEFAULTLSNAME statement.

You need to consider and define:

- ▶ The log stream names and how many log streams to use.
- ▶ If you use DASD-only (no automatically merged) or CF structure log (automatically merged in the CF if more than one system use the same log stream).
- ▶ The use of staging data sets.
- ▶ Structure names.

- ▶ Retention period.
- ▶ Autodelete.
- ▶ System Logger requires that you have SMS installed and active, although the SMF log stream data set and staging data sets do not need to be SMS-managed.

### IXCMIAPU utility example

An example of defining SMF log streams is shown in Figure 7-4 on page 82. In this example, three log streams are defined and shown in Figure 7-4 on page 82. One log stream is DASD-only and the other two are CF structure log streams. For a CF structure log stream, it is necessary to associate the log stream definition with the structure definition as shown in the example. In this example we associated one log stream to one structure, so you can have better control in the CF structure size and the HIGHOFFLOAD threshold; however, you can associate several log streams to one structure.

Consider the following specifications in the examples shown in Figure 7-3.

- ▶ Define the Coupling Facility structures in the CFRM policy couple data set using the IXCMIAPU utility.
- ▶ Review the LOGR CDS (DSEXTENT) definition, to be able to control all the data sets that the SMF log streams create for a long retention period. Defining DSEXTENT(10), allows LOGR to control up to 1680 LOGR data sets for all kinds of log stream data sets. Changing this keyword requires a reallocate of the LOGR CDS.
- ▶ Ensure that there is enough DASD space for the new SMF log stream data set that is dynamically allocated until that space is migrated. If the LOGR policy migrates the file in two days, then you need guarantee DASD space for at least two or three days.

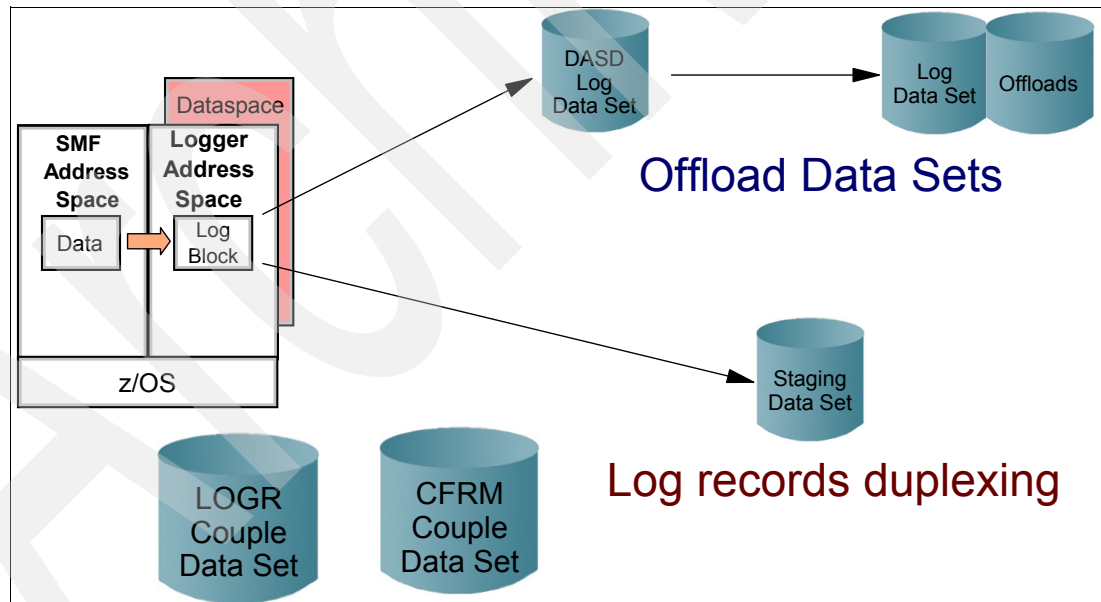


Figure 7-3 System Logger configuration for SMF recording to log streams

Consider the following specifications in the examples shown in Figure 7-4 on page 82.

- ▶ For a DASD-only SMF log stream, you must specify the LOGR policy MAXBUFSIZE parameter to define the maximum log block size, in bytes, that the system can write to the DASD-only log stream.

IBM suggests a MAXBUFSIZE for a DASD-only log stream of 65532. You are required to define a MAXBUFSIZE of at least 33024.



- ▶ For a Coupling Facility log stream, you must specify the LOGR policy MAXBUFSIZE parameter to define the maximum log block size, in bytes, that the system can write to the log stream assigned to the structure you are defining.

IBM suggests that you define a MAXBUFSIZE value between 33024 and 65532. SMF issues an error message if the MAXBUFSIZE value specified is too small.

- ▶ The size of the SMF log stream is defined by the LS\_SIZE keyword. If you define LS\_SIZE(100000), this allocates each data set with 556 cylinders.
- ▶ Defining AUTODELETE(YES) and RETPD(365) for a log stream means that data sets are automatically deleted after 365 days. Defining AUTODELETE(NO) indicates to delete the entries after 365 days using System Logger services which must be provided as a user-written application.
- ▶ HIGHOFFLOAD specifies the percent value you want to use as the high offload threshold for the Coupling Facility space allocated for this log stream. When the Coupling Facility is filled to the high offload threshold point or beyond, System Logger begins offloading data from the Coupling Facility to the DASD log stream data sets. The default HIGHOFFLOAD value is 80%. You can specify the default in the following way:

HIGHOFFLOAD(80)

```

//SMFLOGD JOB 'ACCTNO,ACCTINFO','FIRST LASTNAME',
// MSGLEVEL=(1,1),CLASS=A,NOTIFY=&SYSUID.,MSGCLASS=A
//EXEC1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
DATA TYPE (LOGR)
DEFINE STRUCTURE NAME(SMF_PERF)
LOGSNUM(5)
AVGBUFSIZE(32767)
MAXBUFSIZE(65532)
DEFINE STRUCTURE NAME(SMF_REMIND)
LOGSNUM(5)
AVGBUFSIZE(32767)
MAXBUFSIZE(65532)
DEFINE LOGSTREAM NAME(IFASMF.PERF)
DASDONLY(NO)
STRUCTNAME(SMF_PERF)
STG_DUPLEX(YES)
LS_SIZE(100000)
AUTODELETE(YES)
RETPD(365)
HLQ(LOGR)
HIGHOFFLOAD(80)
LOWOFFLOAD(0)
DEFINE LOGSTREAM NAME(IFASMF.DEFAULT)
DASDONLY(NO)
STRUCTNAME(SMF_REMIND)
STG_DUPLEX(YES)
LS_SIZE(100000)
AUTODELETE(YES)
RETPD(365)
HLQ(LOGR)
HIGHOFFLOAD(80)
LOWOFFLOAD(0)
DEFINE LOGSTREAM NAME(IFASMF.JOB)
DASDONLY(YES)
MAXBUFSIZE(65532)
LS_SIZE(30000)
STG_SIZE(30000)
AUTODELETE(YES)
RETPD(365)
HLQ(LOGR)
HIGHOFFLOAD(80)
LOWOFFLOAD(0)
/*

```

Figure 7-4 JCL example to define an SMF log stream to the LOGR CDS

## 7.2.2 Updating the CFRM policy for SMF CF structure logstream

CFRM policy is defined using IXCMIAPU utility and it is unique in z/OS sysplex; the new SMF structure must be appended to the current policy. You need to get a copy of the last CFRM policy definition using the IXCMIAPU utility and include the statements for the new SMF structures.

Figure 7-5 has an example of the CFRM statements that you need to include in your current policy.

```
STRUCTURE NAME(SMF_PERF)
  SIZE(30000)
  INITSIZE(30000)
  PREFLIST(CF2,CF1)
  REBUILDPERCENT(1)

STRUCTURE NAME(SMF_REMIND)
  SIZE(30000)
  INITSIZE(30000)
  PREFLIST(CF1,CF2)
  REBUILDPERCENT(1)
```

Figure 7-5 CFRM statements to include in policy

After you define a new policy, you must activate it using the following command:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=CFRMPOL1
```

### 7.2.3 Updating the SMFPRMxx parmlib member

During initialization, SMF searches the SMFPRMxx parmlib member to see whether the system is using log streams or SMF data sets to record SMF data. To use the new SMF log stream facility you need to include some new statements in the SMFPRMxx parmlib member.

SMF MANx data sets can still be defined in the SMFPRMxx parmlib member if log streams are to be used. This allows for a fallback to SMF data sets if the exploitation of log streams encounters unexpected problems.

The new keywords are as follows:

```
RECORDING(DATASET | LOGSTREAM)
DEFAULTLSNAME(logstreamname)
LSNAME(logstreamname,TYPE{aa,bb}|({aa,bb:zz}))
```

#### Choosing log stream names

The statement LSNAME specifies the log stream name that will be used for a particular group of SMF records type to be captured by this log stream. You can define a default log stream name on DEFAULTLSNAME, and request particular record types go to particular log streams using the TYPE subparameter on the LSNAME parameter. You can define several LSNAME statements to segregate the SMF records.

```
LSNAME(logstreamname,TYPE({aa,bb}|{aa,bb:zz}))
```

TYPE specifies the SMF record types that SMF is to collect to the specified log stream on the LSNAME parameter. aa, bb, and zz are the decimal notations for each SMF record type. You cannot specify subtypes on the TYPE subparameter for LSNAME. A colon (:) indicates the range of SMF record types (bb through zz) to be recorded.

Value Range: 0-255 (SMF record types)

Default: TYPE (0:255) (all types)

The log stream name must be composed as follows:

- ▶ The first seven characters must be IFASMF. and if the first six characters are not IFASMF, the system issues an error message.
- ▶ You must have a minimum of 8 characters. A log stream name should be a unique descriptive identifier, made up of two or more qualifiers (each 1 to 8 characters in length) separated by periods (.) which you must count as characters.

Each qualifier can contain up to eight numeric, alphabetic, or national (\$, #, or @) characters. The first character of each qualifier must be an alphabetic or national character.

- ▶ You must have a maximum of 26 characters.
- ▶ It must conform to other log stream naming conventions as documented in *IXGINVNT in z/OS MVS Programming: Assembler Services Reference, Volume 2 (IARR2V-XCTLX), SA22-7607*.

**Note:** You can use system symbols and the &SID symbol in SMF log stream names. The resolved substitution text for the &SID system symbol is the system identifier specified on the SID parameter in SMFPRMxx.

Be aware that &SID can be used only to name resources in SMFPRMxx; you cannot specify &SID in other parmlib members.

### Choose SMF recording type

If the member specifies RECORDING(LOGSTREAM), SMF will write the SMF data to the log streams specified on the DEFAULTLSNAME and LSNAME parameters of the SMFPRMxx parmlib member. You can define DATASET or LOGSTREAM. The default is DATASET.

```
RECORDING(DATASET | LOGSTREAM)
```

Although you can identify both SMF data sets and log streams in the SMFPRMxx parmlib member, only one recording mechanism can be in use at a time. Use the following command to easily switch between recording types.

```
SETSMF RECORDING(DATASET|LOGSTREAM)
```

This facilitates both exploitation of the new function, as well as fallback in case of errors.

### Choose default log stream name

The optional statement DEFAULTLSNAME specifies the log stream name that will be used to capture the SMF records type not defined in any of the LSNAME statements. You should define another log stream name for the DEFAULTLSNAME following the same convention as for the of LSNAME statement.

```
DEFAULTLSNAME(IFASMF.SSID.DEFAULT.SMF)
```

### SMF record types to log streams

If you specify the same record type on two or more different LSNAME parameters, the system writes the record to all specified log streams. Figure 7-6 on page 85 shows one example of how to define the new statements in SMFPRMxx. In this case:

- ▶ Record type 30 will be written to both log streams IFASMF.PER and IFASMF.JOB.
- ▶ Record type 89 will be written to the IFASMF.PERF log stream.
- ▶ Record type 04 will be written to the IFASMF.JOB log stream.
- ▶ All the other SMF record types will be written to the IFASMF.DEFAULT log stream.

```
DEFAULTLSNAME (IFASMF.DEFAULT)
LSNAME (IFASMF.PERF,TYPE(30,89))
LSNAME (IFASMF.JOB,TYPE(30,04))
RECORDING (LOGSTREAM)
```

Figure 7-6 SMFPRMxx new statements for log streams

In Figure 7-7, with the use of a colon (:), a range of SMF record types (from 70 to 79) are recorded to log stream IFASMF.PERF and SMF record types 30 and 4 are recorded on log stream IFASMF.JOB. All the other SMF record types will be written to the IFASMF.DEFAULT log stream.

```
DEFAULTLSNAME (IFASMF.DEFAULT)
LSNAME (IFASMF.PERF,TYPE(70:79))
LSNAME (IFASMF.JOB,TYPE(30,04))
RECORDING (LOGSTREAM)
```

Figure 7-7 SMFPRMxx log stream statements segregating RMF records

**Note:** This specification can result in duplicate records being recorded.

## 7.2.4 SMFPRMxx parmlib member considerations

When defining the new statements and keeping the old data set definition in the SMFPRMxx parmlib member, it is possible to switch between the two recording methods using the **SETSMF** command. During an IPL, the recording method statement definition is used or the default is used.

To use the **SETSMF** command to switch between modes, you are required to define the option **PROMPT(LIST)** or **PROMPT(ALL)** in the current SMFPRMxx parmlib member.

It is recommended that you define the **DEFAULTLSNAME** statement for those SMF record types that do not have a specific log stream defined. If a mismatch occurs between the SMF record type that is defined in the **SYS** or **SUBSYS** statement and the SMF record type that has been captured by any **LSNAME** statement, and you are switching from data set mode to log stream mode, then the system will not complete the **SET SMF=XX** command execution and it will continue recording in data set mode.

If you are running with log stream mode and switch to another log stream mode and the mismatch occurs, the system will continue with the old log stream mode until you fix the mismatch and reuse the **SET SMF=XX** command.

**Note:** For SMF log stream recording, you can direct record types to particular log streams by using the **TYPE** subparameter on **LSNAME**. You still select the records you want to write with the **TYPE/NOTYPE** option of **SYS** or **SUBSYS**.

This means it is possible to specify record types on the **TYPE** subparameter of **LSNAME** that the system is not actually recording, because they are not specified on **SYS** or **SUBSYS**.

If you are IPLing your system to run in log stream mode and the mismatch occurs, the system will buffer the SMF records until you fix the problem and reissue the **SET SMF=XX** command.

If a mismatch occurs during a change of the SMFPRMxx parmlib member parameters switch, or an IPL is done, then the following messages may be issued:

```
IFA702I NO LOGSTREAMS WERE SPECIFIED FOR THE FOLLOWING RECORD TYPES n1-nx
IFA710I LOGSTREAM PARAMETERS WILL NOT BE USED DUE TO ERROR
IFA717I LOGSTREAMS ARE NOT USABLE BY SMF. DATA BEING BUFFERED TIME=hh.mm.ss
```

## 7.2.5 Switching to log stream mode

There are several ways to switch to the SMF log stream recording mode:

- ▶ Use the **SETSMF** command if your current SMFPRMxx parmlib member has both (data sets and log stream) definitions and the **PROMPT** keyword.

```
SETSMF RECORDING(LOGSTREAM)
```

- ▶ Use the **SET SMF=xx** command, if you defined a new SMFPRMxx parmlib member with log stream definitions.

```
SET SMF=XX
```

- ▶ When you are IPLing the system and pointing to the new SMFPRMxx parmlib member that specifies **RECORDING(LOGSTREAM)**, however, this is not the preferred way to migrate to SMF log streams. This is because, if you have definition problems, the system will buffer the SMF records in a dataspace until you fix it.

The preferable way to migrate is to create a new SMFPRMxx parmlib member that includes the new statements, and then use the **SET SMF=XX** command because most members use the **NOPROMPT** option. You will receive the messages shown in the IPL example in Figure 7-8, Figure 7-9 on page 87, and Figure 7-10 on page 88.

```
SET SMF=10
IEE252I MEMBER SMFPRM10 FOUND IN SYS1.PARMLIB
IEE967I 10.04.06 SMF PARAMETERS 977
      MEMBER = SMFPRM10
      MULCFUNC -- DEFAULT
      .....
      .....
      SID(SC70) -- DEFAULT
      DEFAULTLSNAME(IFASMF.DEFAULT) -- PARMLIB
      PROMPT(LIST) -- PARMLIB
      DSNAME(SYS1.SC70.MAN3) -- PARMLIB
      DSNAME(SYS1.SC70.MAN2) -- PARMLIB
      DSNAME(SYS1.SC70.MAN1) -- PARMLIB
      ACTIVE -- PARMLIB
*155 IEE357A REPLY WITH SMF VALUES OR U
```

Figure 7-8 Setting the access to a specific SMFPRMxx parmlib member during IPL

```

R 155,U
IEE600I REPLY TO 155 IS;U
.....
IEE974I 10.04.34 SMF DATA SETS 007
      NAME                VOLSER SIZE(BLKS) %FULL STATUS
      P-SYS1.SC70.MAN1    SBOXD5   1500   0 ALTERNATE
      S-SYS1.SC70.MAN2    SBOXD5   1500  67 ACTIVE
      S-SYS1.SC70.MAN3    SBOXD5   1500   0 ALTERNATE
IEF196I IGD100I 6E16 ALLOCATED TO DDNAME SYS00132 DATACLAS (      )
IEF196I IEF237I 6E16 ALLOCATED TO SYS00133
IXC582I STRUCTURE SMF_PERF ALLOCATED BY SIZE/RATIOS. 010
PHYSICAL STRUCTURE VERSION: C09A71D3 A709EB4B
STRUCTURE TYPE:          LIST
CFNAME:                  CF2
ALLOCATION SIZE:          30208 K
POLICY SIZE:             30000 K
POLICY INITSIZE:        30000 K
POLICY MINSIZE:          0 K
IXLCONN STRSIZE:        0 K
ENTRY COUNT:            678
ELEMENT COUNT:          41802
ENTRY:ELEMENT RATIO:    1 :    65
ALLOCATION SIZE IS WITHIN CFRM POLICY DEFINITIONS
IXL014I IXLCONN REQUEST FOR STRUCTURE SMF_PERF 011
WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 0017
CONNECTOR NAME: IXGLOGR_SC70 CFNAME: CF2
IXL015I STRUCTURE ALLOCATION INFORMATION FOR 012
STRUCTURE SMF_PERF, CONNECTOR NAME IXGLOGR_SC70
CFNAME      ALLOCATION STATUS/FAILURE REASON
-----
CF2          STRUCTURE ALLOCATED AC001800
CF1          PREFERRED CF ALREADY SELECTED AC001800
IEF196I IGD100I 836A ALLOCATED TO DDNAME SYS00134 DATACLAS (      )
IEF196I IEF237I 836A ALLOCATED TO SYS00135
IXC582I STRUCTURE SMF_REMIND ALLOCATED BY SIZE/RATIOS. 016
PHYSICAL STRUCTURE VERSION: C09A71D4 33740214
STRUCTURE TYPE:          LIST
CFNAME:                  CF1
ALLOCATION SIZE:          30208 K
POLICY SIZE:             30000 K
POLICY INITSIZE:        30000 K
POLICY MINSIZE:          0 K
IXLCONN STRSIZE:        0 K
ENTRY COUNT:            678
ELEMENT COUNT:          41802
ENTRY:ELEMENT RATIO:    1 :    65
ALLOCATION SIZE IS WITHIN CFRM POLICY DEFINITIONS
IXL014I IXLCONN REQUEST FOR STRUCTURE SMF_REMIND 017
WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 0017
CONNECTOR NAME: IXGLOGR_SC70 CFNAME: CF1
IXL015I STRUCTURE ALLOCATION INFORMATION FOR 018
STRUCTURE SMF_REMIND, CONNECTOR NAME IXGLOGR_SC70

```

Figure 7-9 IPL messages (continued)

```

IFA711I LOGSTREAM PARAMETERS ARE IN EFFECT
IEE536I SMF      VALUE 10 NOW IN EFFECT
D SMF
IFA714I 10.04.36 SMF STATUS 032
      LOGSTREAM NAME          BUFFERS      STATUS
      A-IFASMF.DEFAULT        4325        CONNECTED
      A-IFASMF.PERF           0            CONNECTED
      A-IFASMF.JOB            0            CONNECTED

```

Figure 7-10 Final IPL messages and display of current SMF log stream allocations

## 7.3 Dumping the SMF log stream data set

With z/OS V1R9, a new SMF dump program called IFASMF DL is available. The IFASMF DP utility is used to process the SMF MANx data sets, isolate the records, create files to be post-processed, and clean the MAN data sets. Now, with log stream data sets, the new IFASMF DL utility reads the log streams and can create several different files as required in just one step.

The record data from the new SMF dump program IFASMF DL should be virtually indistinguishable from data dumped by the SMF data set dump program IFASMF DP. You will just need to change your JCL to use the new IFASMF DL utility.

### IFASMF DL dump program utility

The SMF log stream dump program dumps the contents of one or more log streams to sequential data sets on either tape or direct access devices. The SMF log stream dump program allows the installation to route different records to separate files and produce a summary activity report.

A new feature in the SMF log stream dump program is the addition of filters on the OUTDD statements. Previously, each OUTDD would receive all the records specified on the main filter keywords of DATE/START/END/SID. If it was required to create different OUTDD data sets with different records from the same input, it was necessary to run the dump program a second time to create that second data set.

Now, with the new IFASMF DL utility, it is possible to specify those same filters on the OUTDD statement, allowing the capability to read the log stream one time, and partition the output, sending different records to different data sets. This should result in fewer invocations of the dump program being required, in turn causing less impact to the system by reading the log stream only once.

**Note:** A feature in the SMF dump program for log streams allows you to partition output data based on date, time, and SMF ID (SID).

### IFASMF DL utility example

In the example shown in Figure 7-11 on page 89, the three OUTDD statements refer to the three data sets defined in the JCL, and they specify the SMF record types to be written to each data set. The JCL is explained as follows:

- The DCB= keyword has been coded for the output data set defined by OUTDD2. Any block size 4096 or greater may be specified. Choosing a block size suitable for the device type being used will improve storage resource use. For this job, the data set specified by



OUTDD1 will have a system-determined block size. The data set specified by OUTDD2 will have a block size of 32000.

- ▶ The LRECL= keyword has been coded for an output data set defined as OUTDD3. For this job, the data set specified by OUTDD3 will have an LRECL of 32760. For OUTDD1 and OUTDD2, the LRECL will default to 32767.
- ▶ The LSNAME parameters contain the names of three log streams to be dumped.
- ▶ The OUTDD parameters contain filters selecting the SMF record types to be dumped:
  - OUTDD1 specifies that you want to dump record types 0,2,10,15-30, and subtype 1 of record type 33 starting with those issued at 7:30 am and ending at 6:50 pm.
  - OUTDD2 specifies that you want to dump record types 10 through 255 from dates October 1, 2006 through November 30, 2006.
  - OUTDD3 specifies that you want to dump record types 10 through 255.
- ▶ The DATE parameter specifies, for those OUTDD statements which do *not* include the DATE subparameter, that data from January 1, 2006 through December 31, 2006 is to be written.
- ▶ The SID parameters specify that data will be dumped for systems 308A and 308B.

```
//IFASMF DL JOB accounting information
//STEP EXEC PGM=IFASMF DL
//OUTDD1 DD DSN=SMFREC.FEWYPES,DISP=(NEW,CATLG,DELETE)
//OUTDD2 DD DSN=SMF.TYPE10.TYPE255,DISP=(NEW,CATLG,DELETE),
//   DCB=BLKSIZE=32000
//OUTDD3 DD DSN=SMF.TYPE10.TYPE255B,DISP=(NEW,CATLG,DELETE),
//   DCB=LRECL=32760
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
   LSNAME (IFASMF.DEFAULT)
   LSNAME (IFASMF.PERF)
   LSNAME (IFASMF.JOB)
   OUTDD (OUTDD1,TYPE(0,2,10,15:30,33(1)),START(0730),END(1850))
   OUTDD (OUTDD2,TYPE(10:255)),DATE(2006274,2006334)
   OUTDD (OUTDD3,TYPE(10:255))
   DATE (2006001,2006365)
   SID (308A)
   SID (308B)
   END(2400) - DEFAULT
   START(0000) - DEFAULT
```

Figure 7-11 Sample job for dumping SMF log streams

**Note:** There can be any number of input (LSNAME) or output (OUTDD) parameters in the SMF log stream dump program. The log streams are dumped in reverse order.

For example, in Figure 7-11 on page 89, three log streams are specified. After the SMF log stream dump program is processed, the output files contain the records from log stream IFASMF.JOB first, IFASMF.PERF next, followed by the records from IFASMF.DEFAULT.

### IFASMF DL utility output

If you want to use the SMF log stream facility to take advantage of the high speed services that IXGLOGR can provide, and you also want to keep your old way of storing your SMF data,

then you can do it using this new utility. You can define your retention period as long as you need to copy them. The job output will provide the summary activity report shown in Figure 7-12.

SUMMARY ACTIVITY REPORT						
START DATE-TIME 05/17/2007-23:55:41			END DATE-TIME 05/21/2007-12:22:38			
RECORD TYPE	RECORDS READ	PERCENT OF TOTAL	AVG. RECORD LENGTH	MIN. RECORD LENGTH	MAX. RECORD LENGTH	RECORDS WRITTEN
0	2,538	2.14 %	74.00	74	74	1,137
2	0					3
3	0					3
5	45	.04 %	148.64	145	161	0
14	2,367	2.00 %	471.79	344	716	1,612
15	1,457	1.23 %	347.11	344	380	1,400
17	17	.01 %	100.00	100	100	21
18	9	.01 %	144.00	144	144	3
20	360	.30 %	97.09	91	107	485
22	58	.05 %	52.00	52	52	120
23	51	.04 %	134.00	134	134	59
26	78	.07 %	444.98	442	449	0
28	14	.01 %	188.00	188	188	21
32	6	.01 %	320.00	296	416	8
34	6	.01 %	215.00	215	215	8
35	6	.01 %	152.00	152	152	8
40	23,307	19.65 %	76.43	74	890	21,456
41	203	.17 %	332.00	332	332	192
42	12,608	10.63 %	556.93	176	23,796	10,512
60	52	.04 %	592.82	344	691	34
61	32	.03 %	327.18	279	420	30
62	5	.00 %	188.00	188	188	8
64	8	.01 %	458.00	458	458	16
65	23	.02 %	339.26	300	395	14
66	48	.04 %	335.18	286	420	26
71	303	.26 %	1,752.00	1,752	1,752	286
72	29,997	25.29 %	1,177.61	1,076	2,296	28,314
73	304	.26 %	19,704.00	19,704	19,704	286
74	23,048	19.43 %	16,393.68	236	32,720	21,736
75	1,515	1.28 %	264.00	264	264	1,430
78	606	.51 %	11,588.00	1,888	21,288	572
80	1,229	1.04 %	327.36	231	421	780
88	5,508	4.64 %	234.01	161	308	5,172
89	102	.09 %	1,737.76	290	3,146	92
92	12,632	10.65 %	216.05	168	348	10,758
94	50	.04 %	348.00	348	348	48
TOTAL	118,592	100 %	3,731.88	52	32,720	106,650
NUMBER OF RECORDS IN ERROR			0			

Figure 7-12 SYSOUT output from IFASMF DL dump program execution

**Note:** SMF recording to the MANx data sets is retained in this release and the current configuration can remain as it is. In this release, the record data from the new SMF dump program IFASMF DL should be virtually indistinguishable from data dumped by the SMF data set dump program IFASMF DP.

There are expected changes to vendor code that uses “live” SMF data, that is, data that is still resident in the SMF “MAN” data sets. Vendor programs that read data from data sets created by the SMF dump program, as indicated above, are not affected.

### 7.3.1 Using the SWITCH command with log streams

When you need to dump an SMF log stream in order to archive the data to a permanent medium so that an existing user-written analysis routine can run against the dump data sets, you can dump SMF log stream data by issuing the **SWITCH SMF** command. This command first dumps the SMF data in the buffers out to the log streams, and then passes control to the IEFU29L SMF log stream dump exit. Use the SMF log stream dump program IFASMF DL to dump the specified log stream data to dump data sets. For more information about the IEFU29L exit, refer to *z/OS MVS Installation Exits*, SA22-7593.

Note that this differs from the MANx SMF data set dump program, because there is no CLEAR option on the log stream dump program to delete data. When you use log streams to record SMF data, there is no reason to delete data during the dump process. System Logger allows you to manage log data retention using options on the log stream definition in the LOGR couple data set specified using the administrative data utility IXCMIA PU, as shown in Figure 7-4 on page 82.

## 7.4 Migration considerations

SMF MANx data sets can still be defined in the SMFPRMxx parmlib member. This allows a fallback to SMF data sets if the exploitation of log streams encounters unexpected problems.

Although you can identify both SMF data sets and log streams in the SMFPRMxx parmlib member, only one recording mechanism can be in use at a time. The **SETSMF RECORDING (DATASET | LOGSTREAM)** command allows you to easily switch between recording types. This facilitates both exploitation of the new function, and fallback in case of errors.

Consider a situation where you need to change the end location of your data to keep the data completely partitioned throughout (that is, from time of creation to time of archival or deletion). For example, you may not want to change the location of billing data without the concurrence of the data's users. In terms of coexistence, you can use either DASD-only or CF-based log streams. If multiple systems record to the same log stream, be sure to have unique SMF IDs (SIDs) for each system.

### Migration examples

Consider the following ways to migrate from the current SMF recording to the use of log streams and the System Logger.

#### Example 1

You can set up one log stream to write data to a DASD-only log stream, simply replacing SMF MANx data sets. This is probably the simplest approach to using log streams because you will have better performance using a log stream as opposed to using SMF MANx data sets. The benefit of this approach is to familiarize yourself with using the log stream.

Use DEFAULTLSNAME(IFASMF.xxx) or LSNAME(IFASMF.xxx,TYPE(0:255)) to specify the log stream and either use the DEFAULTLSNAME or LSNAME keywords to indicate which log stream to write the records. Then use the new SMF dump program IFASMF DL to extract and dump the records.

## Example 2

You can set up several log streams, with each log stream for a particular purpose:

- ▶ Specify one log stream is to receive all performance-related records, using a log stream named IFASMF.PERF.DATA  
Record types 30, 70-72 and 99
- ▶ Specify a log stream to receive all audit-related records, such as RACF might need, using a log stream named IFASMF.AUDIT.DATA  
Record types 30, 80, 81, and 83
- ▶ Specify a log stream for DB2 data, which can be extensive, named IFASMF.DB2.DATA
- ▶ Specify a log stream for everything else that can be routed to a default log stream for retention or deletion as required

With four separate log streams, four separate SMF dump programs can be used to dump and archive the data. Each dump program can run faster because there are fewer records to process, filter, or ignore.

The retention and offload parameters of each log stream can be tuned via the IXCM2APU utility with TYPE=LOGR to ensure proper retention and performance.

## GRS enhancements

z/OS provides multiple ways of providing serialized access to data on single or multiple systems, but global resource serialization (GRS) is a fundamental way for programs to get the control they need and ensure the integrity of resources in a multisystem environment.

Because global resource serialization is automatically part of a z/OS system and is present during z/OS initialization, it provides the application programming interfaces that are used by the applications on the system. The enhancements are designed to make better use of 64-bit addressing to improve performance for ISGENQ and ENQ/DEQ/RESERVE LINKAGE=SYSTEM users.

This chapter describes the enhancements to global resource scheduling (GRS) in z/OS V1R9:

- ▶ Global resource serialization (GRS) overview
- ▶ Performance enhancements for LATCH processing
- ▶ GRS storage constraint relief
- ▶ ISGECA API support
- ▶ Performance enhancements for CMSEQDQ lock processing

## 8.1 Global resource serialization overview

In a star complex, global resource serialization (GRS) uses an XES lock structure to serialize requests for global resources. All systems in a star complex must be members of the same sysplex and be connected to a Coupling Facility containing the global resource serialization lock structure (ISGLOCK) to manage contention for global resources. For practical purposes, where global resource serialization star complex is concerned, the terms *sysplex* and *complex* are synonymous. No channel-to-channel (CTC) connection of systems, other than those managed by XCF, are supported by global resource serialization in a star complex.

### ISGLOCK lock structure

When a system in a star complex issues an ENQ, DEQ, ISGENQ, or RESERVE request for a global resource, global resource serialization converts the request to a lock request against the ISGLOCK lock structure. Global resource serialization uses the ISGLOCK lock structure to coordinate the requests to ensure proper resource serialization across all systems in the complex. The status of each request is returned to the system that originated the request. Based on the results of these lock requests, global resource serialization will respond to the requester with the outcome of the serialization request.

### Global ENQ/DEQ processing overview

In a star complex, requests for ownership of global resources will be handled through ISGLOCK, the lock structure, on a Coupling Facility that is fully connected with all the systems in the sysplex. Global resource serialization uses the ISGLOCK lock structure to reflect a composite system-level view of the interest in every global resource, for which there is at least one requester. In general, global resource serialization alters this composite view each time you change the set of requesters for the resource.

Each time an ENQ request is received, global resource serialization processing analyzes the state of the resource request queue for the resource. If the new request alters the composite state of the queue, an IXLLOCK macro request is made to reflect the changed state of the resource for the requesting system. If the resource is immediately available, the requester is then granted ownership of the resource.

If the resource is not immediately available, global resource serialization will maintain the request in the waiting state. When the appropriate DEQ request is received, global resource serialization will either resume or post the requester (depending on the ENQ options).

### 8.1.1 Setting address space ENQ limits

GRS enforces a limit on the maximum number of concurrent ENQ, ISGENQ, RESERVE, GQSCAN, and ISGQUERY requests issued by an address space. The general purpose of enforcing a limit is to prevent a runaway ENQ address space from exhausting GRS private storage. In z/OS releases prior to V1R8, the maximum number of concurrent requests is:

- ▶ 4,096 for unauthorized requests
- ▶ 250,000 for authorized requests

These are system-wide maximums. They apply to all address spaces in the system. Note that ENQ RET=CHNG, ENQ RET=TEST, and their equivalent ISGENQ requests do not increase the number of concurrent requests. However, GQSCAN and ISGQUERY REQINFO=QSCAN requests, where the filled answer area results in a request token for continuation, *do* increase the number of concurrent requests.

## Current application ENQ problems

As installations grow and applications get more complex, the number of concurrent ENQ requests may rise to exceed the maximum values. An unauthorized workload such as CICS may already exceed the current maximum values in your environment. Future authorized workloads such as DB2 V8 may pose a similar problem. Some installations are already zapping the maximum values to allow for a greater number of concurrent ENQs than the system-wide defaults.

## New limits for ENQs

GRS is enhanced in z/OS V1R8 to allow greater flexibility and control over the system-wide maximums. In z/OS V1R8, the default values are:

- ▶ 16,384 for unauthorized requests
- ▶ 250,000 for authorized requests

z/OS V1R8 allows you to set the system-wide maximum values. Furthermore, it is now possible for an authorized caller to set its own address space-specific limits. These new values substantially increased STAR mode capacity. The majority of the persistent ENQs are global ENQs as they are data set related. However, z/OS V1R8 only provided minimal relief for GRS=NONE, GRS=RING mode systems, and systems running some ISV serialization products.

## Current installation modifications

In order to provide support for setting the maximums, some non-programming interfaces with the z/OS V1R8 are changed. If you are already zapping the maximums at your installation (a non-programming interface), the changes in z/OS V1R8 may cause some incompatibilities. Prior to z/OS V1R8, it was necessary to zap the GVTCREQ and GVTCREQA fields in the GVT to change the maximums. When you upgrade to z/OS V1R8, there is a new mechanisms for setting the maximums.

**Note:** The GVTCREQ and GVTCREQA fields are no longer zappable with z/OS V1R8.

To check whether you are currently zapping the maximums in the GVT at your installation, follow this procedure (remember this is only true for pre-z/OS V1R8 releases):

- ▶ From ISPF, you can use ISRDDN as follows:  
ISRDDN provides a BROWSE command for browsing storage and loaded modules. We use the BROWSE command to check the values in the GVT fields.

- ▶ From the ISRDDN command line, enter this command:

```
BROWSE 0.+10?+1B0?
```

This command uses the pointer at offset X'1B0' of the CVT to get to the GVT. The pointer to the CVT is located at offset X'10' in virtual storage. After issuing the command, a browse screen is displayed.

- ▶ To verify you are really looking at the GVT, check for the "GVT" eye catcher at offset X'0', as shown in Figure 8-1 on page 96.
- ▶ Check the GVTCREQ field value at offset X'80' and the GVTCREQA field value at offset 84.

If you see the default values in the GVTCREQ and GVTCREQA fields, then you are not zapping the GVT in your installation. When using the default values, the GVTCREQ field holds a value of 4,096 (X'1000') and the GVTCREQA field holds a value of 250,000 (X'3D090'), as displayed in Figure 18-1 on page 376.

- ▶ Check the GVTCREQ field value at offset X'80' and the GVTCREQA field value at offset 84.

If you see the default values in the GVTCREQ and GVTCREQA fields, then you are not zapping the GVT in your installation.

**Note:** When using the default values, the GVTCREQ field and the GVTCREQA field holds a value of 250,000 (X'3D090'), as displayed in Figure 8-1 on page 96.

```

BROWSE      STORAGE      Start:00FD8830                               Line 00000000 0.+10?+1B0?
Command ==>                               Scroll ==> PAGE
***** Top of Data *****
+0 (00FD8830) C7E5E340 B2000800 09100000 C3C9F0F1 * GVT ¥.....CI01 *
+10 (00FD8840) 007C3000 00F80E80 00000000 02720580 * .@...8.0....É.0 *
+20 (00FD8850) 80000000 01810068 007FD0C8 00000000 * 0....a.Ç."}H... *
+30 (00FD8860) 000005C0 00001194 00000000 7E4A0000 * ...{...m....=ç.. *
+40 (00FD8870) 00000000 00000000 00000000 00000000 * ..... *
+50 (00FD8880) 00F4B860 02702000 00FBA000 00F4BD10 * .4%-o...0µ..4" *
+60 (00FD8890) 02A44076 02A4401C 02A4426B 00030000 * .u î.u ..uâ,... *
+70 (00FD88A0) 00F801F8 10001000 01F00000 00000000 * .8.8....0..... *
+80 (00FD88B0) 00001000 0003D090 00FD8B28 00000000 * .....}°.Ù».... *
+90 (00FD88C0) E2C3F6F3 40404040 00000001 00000000 * SC63..... *
+A0 (00FD88D0) 00000000 00000000 00000000 00000000 * ..... *
+B0 (00FD88E0) 00000000 00000000 00000000 00000000 * ..... *
+C0 (00FD88F0) 00000112 24567700 000A07FE 07FE0000 * .....îÿ....Û.Û.. *
+D0 (00FD8900) 00000000 0002BF20 0000000A 0000000A * .....x..... *
+E0 (00FD8910) 00000014 00000014 00000064 0000000A * .....À.... *
+F0 (00FD8920) 00000014 00000050 00000064 000000C8 * .....8...Â...H *
+100 (00FD8930) 00000014 00028000 00000005 80000000 * .....0....0... *

```

Figure 8-1 Verifying GVT fields using the ISRDDN BROWSE command

### ISGADMIN service

The ISGADMIN service allows you to programmatically change the maximum number of concurrent ENQ, ISGENQ, RESERVE, GQSCAN, and ISGQUERY requests in an address space. This is useful for subsystems such as CICS and DB2, which have large numbers of concurrently outstanding ENQs, query requests, or both. Using ISGADMIN, you can set the maximum limits of unauthorized and authorized concurrent requests. It is impossible to set the maximums lower than the system-wide default values.

### New keywords in the GRSCNFxx parmlib member

To allow you to control the system-wide maximums, two new keywords are added to the GRSCNFxx parmlib member:

- ENQMAXU(value)** Identifies the system-wide maximum of concurrent ENQ requests for unauthorized requesters. The ENQMAXU range is 16,384 to 99,999,999. The default is 16,384.
- ENQMAXA(value)** Identifies the system-wide maximum of concurrent ENQ requests for authorized requesters. The ENQMAXA range is 250,000 to 99,999,999. The default is 250,000.

### SETGRS ENQMAXU and SETGRS ENQMAXA operator commands

z/OS V1R8 provides operator commands to allow you to dynamically set the system-wide maximums. Use the **SETGRS ENQMAXU** command to set the system-wide maximum number of concurrent unauthorized requests, and the **SETGRS ENQMAXA** command to set the system-wide maximum number of concurrent authorized requests.

The format of the new operator commands is shown in Figure 8-2 on page 97.



```
SETGRS ENQMAXU=nnnnnnnn[,NOPROMPTINP]  
SETGRS ENQMAXA=nnnnnnnn[,NOPROMPTINP]
```

Figure 8-2 Format of the SETGRS ENQMAXU and ENQMAXA operator commands

### GRS messages for concurrent requests

When an address space exceeds 80% of the maximum concurrent requests, GRS issues the ISG368E message. When the number of concurrent requests in the address space drops below 75% of the maximum, the ISG369I message is issued and the ISG368E message is DOMed. You can use the NOPROMPT parameter to automate the response to message ISG368E.

In general, the SETGRS ENQMAXU and SETGRS ENQMAXA commands are intended for emergency situations and not for normal operations. Keep in mind that the system-wide maximums are intended to protect GRS from runaway ENQ address spaces. Increasing the system-wide maximums makes GRS more vulnerable to run away ENQ address spaces.

If you have an address space that requires a higher maximum than the system-wide maximums, the recommended action is to change the application to use the ISGADMIN service to request a higher address space-specific maximum.

## 8.1.2 Contention notification system movement

Resource contention can result in poor system performance. When resource contention lasts over a long period of time, it can result in program starvation or deadlock conditions.

When running in ring mode, each system in the GRS complex is aware of the complex-wide ENQs, which allows each system to issue the appropriate ENF 51 contention notification event. However, when running in star mode, each system only knows the ENQs that are issued by itself.

To ensure that the ENF 51 contention notification event is issued on all systems in the GRS complex in a proper sequential order, one system in the complex is appointed as the sysplex-wide contention notifying system (CNS). All ENQ contention events are sent to the CNS, which then issues a sysplex wide ENF 51.

GRS provides two APIs to query for contention information:

**ISGECA** Obtains waiter and blocker information for GRS managed resources.

**ISGQUERY** Obtains the status of resources and requester of those resources.

GRS issues an event notification facility (ENF) signal 51 to notify monitoring programs to track resource contention. This is useful in determining contention bottlenecks, preventing bottlenecks, and potentially automating correction of these conditions.

During an IPL, or if the CNS can no longer perform its duties, any system in the complex can act as the CNS. You can determine which system is the current CNS with the D GRS command. In the example shown in Figure 8-3 on page 98, you can see that the CNS is SC64.

```

D GRS
ISG343I 15.44.37 GRS STATUS 544
SYSTEM   STATE           SYSTEM   STATE
SC65    CONNECTED        SC64    CONNECTED
SC70    CONNECTED        SC63    CONNECTED
GRS STAR MODE INFORMATION
LOCK STRUCTURE (ISGLOCK) CONTAINS 1048576 LOCKS.
THE CONTENTION NOTIFYING SYSTEM IS SC64
SYNCHRES:    YES
ENQMAXU:    16384
ENQMAXA:    250000
GRSQ:    CONTENTION

```

Figure 8-3 Output of the DISPLAY GRS command

### The SETGRS CNS operator command

The CNS can take up a considerable amount of system resources. You may prefer that it does not reside on your main production system, or you may prefer that it does not reside on a test system with insufficient capacity to handle the function. Starting with z/OS V1R8, you can choose the system to act as the CNS with the SETGRS CNS command. The format of this command is shown in Figure 8-4.

```
SETGRS CNS=system-name[,NOPROMPT|NP]
```

Figure 8-4 Format of the SETGRS CNS command

Figure 8-5 shows an example of changing the CNS from system SC64 to system SC70.

```

SETGRS CNS=SC70
*092 ISG366D CONFIRM REQUEST TO MIGRATE THE CNS TO SC70. REPLY CNS=SC70
  TO CONFIRM OR C TO CANCEL.
092CNS=SC70
IEE600I REPLY TO 092 IS;CNS=SC70
IEE712I SETGRS PROCESSING COMPLETE
ISG364I CONTENTION NOTIFYING SYSTEM MOVED FROM SYSTEM SC64 TO SYSTEM SC70.
  OPERATOR COMMAND INITIATED.

D GRS
ISG343I 15.54.47 GRS STATUS 565
SYSTEM   STATE           SYSTEM   STATE
SC65    CONNECTED        SC64    CONNECTED
SC70    CONNECTED        SC63    CONNECTED
GRS STAR MODE INFORMATION
LOCK STRUCTURE (ISGLOCK) CONTAINS 1048576 LOCKS.
THE CONTENTION NOTIFYING SYSTEM IS SC70
SYNCHRES:    YES
ENQMAXU:    16384
ENQMAXA:    250000
GRSQ:    CONTENTION

```

Figure 8-5 Setting the CNS using SETGRS command

### Setting CNS during the IPL

It is currently not possible to set the CNS from the GRSCNFxx parmlib member, because GRSCNFxx parsing is done too early in system initialization for setting the CNS across the sysplex. If you require the CNS to reside on a specific system, it is recommended that you use the SETGRS CNS operator command with the NOPROMPT parameter. You can add the

command to the COMMNDxx parmlib member to have it issued automatically at IPL. An example of the SETGRS CNS command with NOPROMPT is shown in Figure 8-6.

```
SETGRS CNS=SC64,NP
IEE712I SETGRS PROCESSING COMPLETE
ISG364I CONTENTION NOTIFYING SYSTEM MOVED FROM SYSTEM SC70 TO SYSTEM SC64.
OPERATOR COMMAND INITIATED.
D GRS
ISG343I 16.04.36 GRS STATUS 746
SYSTEM STATE SYSTEM STATE
SC65 CONNECTED SC64 CONNECTED
SC70 CONNECTED SC63 CONNECTED
GRS STAR MODE INFORMATION
LOCK STRUCTURE (ISGLOCK) CONTAINS 1048576 LOCKS.
THE CONTENTION NOTIFYING SYSTEM IS SC64
SYNCHRES: YES
ENQMAXU: 16384
ENQMAXA: 250000
GRSQ: CONTENTION
```

Figure 8-6 Setting the CNS using the SETGRS command with NOPROMPT

### Coexistence with down-level systems

The support for CNS movement across systems in a GRS star complex requires that all systems in the complex run z/OS V1R8. Systems running z/OS V1R7 with APAR OA11382 are also supported. If any system in the complex is an earlier release than z/OS V1R7, the SETGRS CNS command cannot be issued by any member of the complex.

If the CNS system fails, one of the remaining systems in the complex automatically becomes the new CNS. You can use automation for the rare case when the CNS moves from one system to another. Depending on the trigger (operator command or system failure), the system issues one of the following messages in case the CNS moves from one system to another:

```
ISG364I CONTENTION NOTIFYING SYSTEM MOVED FROM SYSTEM xxxx TO SYSTEM yyyy.
OPERATOR COMMAND INITIATED.
```

```
ISG364I CONTENTION NOTIFYING SYSTEM MOVED FROM SYSTEM xxxx TO SYSTEM yyyy.
SYSTEM INITIATED.
```

## 8.2 GRS storage constraint relief with z/OS V1R9

GRS has its own internal storage management system to make the best overall use of its private storage and maintain performance for its various services. In systems with many data sets left open for long periods of time, it is possible to exhaust storage. If left unchanged, this problem would only worsen as system capacity increases.

z/OS V1R9 provides storage constraint relief for all GRS modes by using 64-bit addressing for storage used to represent all ENQs in all GRS modes. Users that keep more than 100 k data sets allocated have exhausted the GRS private area to represent all these ENQs. All the control blocks now reside in 64-bit mode.

The enhancements for this new support in z/OS V1R9 have the following new interfaces:

- ▶ A new MOVEWAITER function for the existing ISGADMIN service is being added for specific queue manipulation required by ISVs.
- ▶ A new resource instance token for resources local to the current system is presented to some ISV-oriented installation exits and also returned by ISGQUERY. This allows the ISV to be able to coordinate events between exits.
- ▶ The ENQTOKEN representing the instance of an ENQ requester's interest in the resource (a QEL) is presented to the queued (ISGNQXITQUEUED1) exit.

During the rewrite of these enhancements, the following improvements were made that provide better performance for ENQ intensive environments and better serviceability:

- ▶ The quality of dump contents improves, which helps to insure better first failure data capture.
- ▶ A significantly reduced CMSEQDQ lock hold time will be experienced.
- ▶ Much better performance should be seen for ISGENQ and ENQ/DEQ/RESERVE LINKAGE=SYSTEM users.
- ▶ A detailed system trace is now issued for all ENQ-related services.
- ▶ The time that an ENQ/RESERVE/ISGENQ is issued is externalized via GRS IPCS reports and ISGQUERY. The ENQ time can be useful for problem determination or run time checks. It has been captured by GRS in previous releases, but had not been externalized.

**Note:** These enhancements should provide a major increase in capacity for all GRS modes. It should be noted that GRS RING is not designed for large ENQ capacity and as such will perform poorly with many outstanding ENQs. Installations should migrate to GRS STAR if running a RING that is planned to support a large number of ENQs. This enhancement provides ENQ storage constraint relief and should provide better DB2 performance because more data sets can be concurrently opened.

Some serialization products reference and sometimes alter GRS control blocks. With z/OS V1R8, to access these control blocks, GRS provides APIs as required so they no longer need to directly reference GRS blocks. Now with z/OS V1R9, ISGECA, ENF 51, or ISGQUERY may be used as an alternative to run GRS queue control blocks.

### **Discontinue using the ISGERQA parameter in with z/OS V1R9**

ISGERQA is a non-documented parameter in the DIAGxx parmlib member that could be used under IBM service recommendation to control the amount of virtual storage below the 2 GB bar that GRS reserves for the extended resource queue area (ERQA). The ERQA is used mainly for ENQ-related control blocks and in z/OS V1R9, they were moved above the 2 GB bar.

**Note:** If this parameter has been defined previously, you must delete it so that this area can be used for others GRS functions.

SYS1.PARMLIB(DIAGxx) keyword ISGERQA is an undocumented control that can increase the size of the GRS below the bar control block storage area (ERQA). It has been used by various installations to increase the GRS ENQ capacity on older releases. However, it is only to be used with Level2 support assistance because it can reduce the amount of other virtual storage available to the GRS address space.

If you are sharing the parmlib with down-level systems, then isolate the DIAGxx ISGERQA to those systems only. The reason for this is because the ERQA still exists and the DIAGxx ISGERQA is still supported and the reason for previously increasing the ERQA has been eliminated (all ENQ-related persistent blocks are now above the bar). The below the bar virtual that the ERQA is consuming can be much smaller. The new GRS defaults will insure that the GRS address space has enough below the bar virtual for other required functions.

## 8.2.1 Ensure that GRSCNFxx is used properly for GRS=NONE

Prior to z/OS V1R8, the GRSCNFxx parmlib member was not processed when global resource serialization was operating in NONE mode. Starting with z/OS V1R8, GRSCNFxx is parsed when IEASYSxx keyword GRS is set to NONE. If you specify GRS=NONE and a GRSCNFxx member, the GRSCNFxx member must now be syntactically correct.

In addition, the following keywords are now relevant for all modes including GRS=NONE:

- ▶ SYNCHRES
- ▶ CTRACE
- ▶ ENQMAXA
- ▶ ENQMAXU

### Step verification

To avoid warning messages and to ensure proper function in GRS=NONE mode, ensure that the GRSCNFxx member is accessible and contains the correct configuration parameters for a GRS=NONE mode system. If global resource serialization is not active and GRSCNFxx is not accessible, the following messages can be issued:

- ▶ The system issues message ISG313I when mode-irrelevant keywords are found in GRSCNFxx parsing.
- ▶ The system issues WTOR message ISG163D when GRSCNFxx cannot be accessed. If you press Enter for this WTOR message, the system continues to issue message ISG372E to indicate that the GRSDEF defaults are used.

GRSCNF00 is the parmlib member selected if you do not specify GRSCNF=xx in parmlib member IEASYSxx. A copy of GRSCNF00 is shipped with the system, but it might need customizing.

**Note:** APAR OA11382 rolled back this facility to z/OS V1R7 and some facility to z/OS V1R8, so this APAR should be applied for both previous releases.

## 8.2.2 GRS exit routines in cross-memory mode

In previous releases, the exit routines ISGNQXITBATCH, ISGNQXITBATCHCND, ISGNQXITPREBATCH, and ISGNQXITQUEUED1 were called in non-cross-memory mode. With z/OS V1R9, these exits are called in cross-memory mode.

If your installation uses any of these exit routines, or if any of your ISV software uses these exit routines, you must ensure that the exit routines run in a cross-memory environment prior to implementing z/OS V1R9. A failure in any of these exit routines could cause a data integrity problem or system failure.

### Step verification

Determine whether any of the affected exit routines are currently in use on your system by using the **DISPLAY PROG,EXIT** command, as shown of Figure 8-7 on page 102.

If you are using any of the affected exit routines, do the following:

- ▶ If any of the exit routines are owned by ISVs that you did not contact for new z/OS V1R9 support, contact those ISVs to ensure that you have the latest updates.
- ▶ If any of the exit routines are owned by your installation, ensure that these exit routines have been modified to enable execution in a cross-memory environment.

Figure 8-7 displays the D PROG,EXIT output.

D PROG,EXIT				
CSV460I 14.58.17 PROG,EXIT DISPLAY 418				
EXIT	DEF	EXIT	DEF	EXIT
ISGNQXITFAST	E	SYS.IEFACTRT	E	SYSSTC.IEFACTRT
SYS.IEFUJI	E	SYSSTC.IEFUJI	E	SYS.IEFU83
SYSSTC.IEFU83	E	CSVDYLPA	E	CSVDYNEX
HZSADDCHECK	E	IEASDUMP.QUERY	E	IEASDUMP.GLOBAL
IEASDUMP.LOCAL	E	IEASDUMP.SERVER	E	IXC_ELEM_RESTART
IXC_WORK_RESTART	E	ISGNQXIT	E	ISGCNFXITSYSTEM
ISGCNFXITSYSPLEX	E	<b>ISGNQXITBATCH</b>	E	<b>ISGNQXITQUEUED1</b>
ISGENDOFLQCB	E	<b>ISGNQXITPREBATCH</b>	E	<b>ISGNQXITBATCHCND</b>
ISGDGRSRES	E	CNZ_MSGTOSYSLOG	E	IEHINITT_EXIT
IEF_ALLC_OFFLN	E	IEF_SPEC_WAIT	E	IEF_VOLUME_ENQ
IEF_VOLUME_MNT	E	IEFDB401	E	CEE_ABEND_EXIT
CNZ_WTOMBEXIT	E	IEFJFRQ	E	SYSSTC.IEFUSO
SYSSTC.IEFUJP	E	SYSSTC.IEFU85	E	SYSSTC.IEFU84
SYSSTC.IEFU29	E	SYS.IEFU29	E	SYS.IEFUTL
SYS.IEFUSO	E	SYS.IEFUJP	E	SYS.IEFUSI
SYS.IEFUJV	E	SYS.IEFU85	E	SYS.IEFU84
IRREXV01	E	IGDACSDX	E	BPX_PREPROC_INIT
BPX_POSPROC_INIT	E	BPX_IMAGE_INIT	E	BPX_PREPROC_TERM
HASP.\$EXITO	E	CBRUXTVS_EXIT	E	

Figure 8-7 D PROG,EXIT output

### ISGNQXIT EQDQ exit

The installation-oriented EQDQ exit environment is not being changed. If using the ISGNQXIT EQDQ exit installed, then consider converting it to the ISGNQXITFAST exit because it is a better performer. However, unlike the ISGNQXIT exit, the ISGNQXITFAST exit can be called in a cross-memory environment.

## 8.2.3 ISGADMIN enhancement

A new function known as MOVEWAITER was introduced in the ISGADMIN services. It allows one application to move one ENQ WAITER to another position in the queue and optionally changes its control type via the NEWCONTROL keyword.

**Note:** This function is intended to only be used by third-party serialization products. Its misuse can result in deadlocks, incorrect serialization, or loss of data integrity.

## 8.3 GRS performance enhancements with z/OS V1R9

As CPU power has increased over time, more ASIDs are created and installations have noted an increase in GRS CPU consumption time, mainly in a sysplex environment. In z/OS V1R9, changes have been effected in order to obtain better performance with GRS processing.

The cost of a hardware cache miss has been increasing over machine generations. To reduce cache misses off a free queue, it is better to keep a separate queue per CPU on its own cache boundary and then have each CPU only take elements off its queue. This technique has shown to provide performance improvements.

The GRS latch function has been modified to use such an algorithm for its free queues and to use the new multi-header CPOOL (one header per CPU) support for some of its CPOOL service-managed storage pools. This benefits all users of GRS latch including z/OS UNIX, RACF, System Logger, IOS, and ISV software.

### GRS latch support

Because the multi-header technique can increase storage demands in the latch set creator address space, GRS latch support provides the following additional functions:

- ▶ A new ISGLCRT\_LOWSTGUSAGE latch set create option to allow latch owners to force the old algorithm to be used because they know their address space is storage-constrained. To date, no one is known to have used this function. The default is to use the multi-header approach.
- ▶ The ability for the GRS latch function to detect and react to a storage constraint in the latch set creator's address space. When constrained, GRS latch will revert the latch set back to the old.

To support the movement of most of the control blocks above the bar, several GRS modules are rewritten to provide better performance in the following areas:

- ▶ Reduce contention on the internal CMSEQDQ lock
- ▶ Reduce path length for ISGENQ and ENQ/DEQ/RESERVE LINKAGE=SYSTEM
- ▶ LATCH processing

**Note:** To execute one instruction, all referenced storage must be in the L1 cache of the processor. If the execution of one latch request starts in one processor and ends in another, then the references to these storages in the second processor cause a cache missing.

This new performance algorithm is the default in z/OS V1R9, and it can cause a slight increase in the latch set creator's private storage usage. In the unlikely case that the space is constrained, latch processing will revert back to the old algorithm automatically. Applications that know their address space storage constraints can force the old algorithm by using a new ISGLCRT\_LOWSTGUSAGE latch set create option.

## 8.4 GRS debugging improvements

A new system trace was added in z/OS V1R8 for PC ENQ for LINKAGE=SYSTEM or ISGENQ. The PC, SSRV (new trace) and PR (return) can be used to determine the call sequence. In z/OS V1R9, the trace is now cut for any type of ISQENQ, ENQ/RESERVE, or DEQ for LINKAGE=SYSTEM for SVC ENQs. The SSRV trace contains detailed information

on the return address, QNAME, return code, scope, disposition, and if RNLs or exits changed request.

### System trace entries

GRS has provided system trace entries for the execution of ENQ/RESERVE (SVC X'38') and DEQ (SVC X'30') forever.

Now, GRS creates one system entry for any execution of the ENQ/RESERVE, DEQ/RELEASE and ISGENQ, executing in SVC mode, LINKAGE=SYSTEM or PC mode. As a result, you can now determine the true execution sequence in the trace.

### SSRV trace entries

The SSRV system trace entries provides many details on the return address:

- ▶ What was requested by issuer
  - QNAME/UCB - RNAME is not provided
  - Disposition: Exclusive/Shared request
  - Request scope: STEP, SYSTEM, SYSTEMS
  - Was a reserve issued
  - What RET was specified: NONE, HAVE, CHNG, USE, ECB, or TEST
  - Last element of list issued
  - RNL=YES/NO
  - Is the TCB specified
- ▶ What change was complemented or attempted
  - An exit changed the request
  - RNLs changed the scope
  - Included, excluded, and converted bits to show how change occurred
  - Final scope issued
  - If a reserve, the UCB address is provided
- ▶ Final result
  - Return or abend code issued

### IPCS VERBX GRSTRACE

With z/OS V1R8, the IPCS VERBX GRSTRACE command formats the time stamp information and the request type. Also, the resource creation time stamp is informed; that is, for how long a period you could have a contention in this resource.

With z/OS V1R9, the IPCS VERBX GRSTRACE command formats the status of each ENQ request, if it is the:

- ▶ /OWN = owner
- ▶ /WAIT = waiter
- ▶ /USE = indicates a MASID request where the user is considered an owner via the MASID target

When using the ENQ or RESERVE requests from authorized callers, use the MASID (matching ASID) and MTCB (matching TCB) parameters to allow a further conditional control of a resource.

One task issues an ENQ or RESERVE for a resource specifying a matching ASID. If the issuing task does not receive control, it is notified whether the matching task has control (which allows the issuing task to use the resource even though it could not acquire the resource itself). This process requires serialization between the issuing and requesting tasks.



## Message Flood Automation

Many z/OS systems are troubled by cases where a user program or a z/OS process itself issues a large number of messages to the z/OS consoles in a short time. Cases of hundreds (or even thousands) of messages a second are not uncommon. These messages are often very similar or identical, but are not necessarily so. Techniques to identify similar messages can be very difficult and time-consuming.

Message Flood Automation addresses this problem. This implementation does not claim to identify all cases of erroneous behavior, or to take the “correct” action in all cases. Its intention is to identify runaway WTO conditions that can cause severe disruptions to z/OS operation, and to take installation-specified actions in these cases.

The problem that Message Flood Automation is attempting to solve has been with the operating system since the earliest days. It is most often caused by malfunctioning input or output devices that flood the system with a very large number of error messages, usually within a matter of seconds. Message floods can also be caused unintentionally by improperly designed programs that get stuck in a message loop, as well as by malicious programs that deliberately generate large volumes of messages in an attempt to cause a system outage.

Message floods are a concern because they monopolize system resources and prevent the operator (and automation) from seeing and reacting to other system messages in a timely manner. In some cases, message floods so monopolize system resources that resource shortages develop and a system outage occurs.

This chapter provides an overview of the Message Flood Automation that was incorporated into z/OS Version 1 Release 9. In this chapter, the following topics are described:

- ▶ Message Flood Automation overview
- ▶ Problem statement and solution
- ▶ Installation, loading and activating
- ▶ Customization and tuning
- ▶ Operator commands

## 9.1 Message Flood Automation overview

Message Flood Automation is a new component of z/OS console support that was made available as a small programming enhancement (SPE) for z/OS Version 1 Release 6, Release 7 and Release 8 at the end of November 2006. Message Flood Automation is being incorporated into z/OS Version 1 Release 9. The parent APAR is OA17514, and the SPE is shipped as follows:

- ▶ Release 709 (z/OS v1R6): UA30810 available 06/11/29 (F611)
- ▶ Release 720 (z/OS V1R7): UA30811 available 06/11/29 (F611)
- ▶ Release 730 (z/OS V1R8): UA30812 available 06/11/29 (F611)

If the Message Flood Automation function is being used prior to z/OS V1R9, the relevant documentation is available in a user's guide which can be found at the following link:

<http://publibz.boulder.ibm.com/zoslib/pdf/mfaguide.pdf>

You are strongly encouraged to read the user's guide fully before attempting to set up and activate Message Flood Automation.

**Note:** If you have already been running Message Flood Automation and are now installing a newer level, read the topic "Considerations when migrating from one level to another" in *Message Flood Automation User's Guide*—for z/OS releases prior to z/OS V1R9.

With z/OS V1R9, documentation is available with the standard manuals.

## 9.2 Message Flood Automation implementation

Message Flood Automation is designed to detect and react to a message flood before those consequences have had an opportunity to occur. A defined installation policy allows installations to tailor Message Flood Automation to an individual environment by deciding how quickly the Message Flood Automation should react to a potential message flood and then, what actions should be taken if a message flood occurs. Depending on the policy that is established, Message Flood Automation can prevent message buffers from filling, console queues from becoming overly long, and console displays from becoming unreadable.

Because Message Flood Automation deals with the flood messages as they are being generated, large numbers of flood messages do not have the opportunity to accumulate in message buffers or in various queues. This means that there is no need to take action, either automated or manual, to "flush" these unwanted messages from buffers or queues. (In past releases, this was one of the more serious aspects of dealing with message floods, that is, flushing unwanted messages from the queues of each console.)

Furthermore, Message Flood Automation deals with the messages going to all of the various consoles, including the EMCS consoles used by SDSF and by automation products such as Tivoli NetView and Tivoli System Automation.

## 9.2.1 Message flood problems

Message Flood Automation is a new implementation designed to handle disruptions that are caused due to the following situations:

- ▶ Large numbers of messages to the z/OS consoles can obscure important messages and delay them from being acted on.
- ▶ Large numbers of messages to the automation system can delay the processing of normal messages.
- ▶ Messages can use excessive CPU and storage resources. Buffering excessive message traffic may use large amounts of virtual and real storage, and can cause SQA to overflow into CSA. This can cause jobs, subsystems, and even complete systems to be delayed or fail.

Messages can be produced at very high volumes due to:

- ▶ Malfunctioning I/O devices such as DASD, DASD controllers
- ▶ ESCON/FICON switches
- ▶ Malfunctioning network devices
- ▶ Errant programs (unintentional)
- ▶ Malicious programs

Message Flood Automation can react to potential message flooding situations in a matter of tens or hundreds of messages (specifiable by the installation), well before buffers have begun to fill, well before console queues have begun to build, and well before console message rates have begun to become enormous.

Furthermore, its actions do not result in residual buffers or queues of messages that must be “worked down” to return to normal processing. Because its processing is targeted to the messages that are causing the problem, very few uninvolved messages are acted upon.

By contrast, the act of flushing console queues (with the **K Q** command) can result in throwing away many innocent and often important messages. Message Flood Automation can potentially eliminate the need to issue the **K Q** command by preventing flood messages from ever reaching a console. Message Flood Automation has a policy that allows installations to target individual messages, individual jobs, or started tasks. By targeting specific messages and units of work, Message Flood Automation is able to minimize the disruption to other work in the system.

## 9.2.2 MPF processing

The message processing facility (MPF) controls message processing for an MVS system. Message Flood Automation runs as part of MPF processing, which occurs after the control block that represents the message has been created. Message Flood Automation is able to see and alter any processing of the message that occurs prior to the creation of this control block.

**Note:** Some automation products replace the Write-To-Operator (WTO) Supervisor Call (SVC) with their own code and then invoke the WTO code when they are finished. Message Flood Automation is able to see and react to messages that have been “front-ended” by other automation in this way.

### MPFLSTxx parmlib member

The IEAVMXIT installation exit or an MPF installation exit (one that you specify on the USEREXIT parameter in an MPFLSTxx parmlib member) allows you to modify message processing in a system or sysplex. IEAVMXIT is the general-purpose exit routine that performs processing that is common to many messages (WTOs). See 9.3.1, “Message Flood Automation exits” on page 109 an explanation about the detailed use of this exit with Message Flood Automation support.

Because all messages are processed by MPF, the MPFLSTxx parmlib member tells MPF what to do with each message. The following is a list of changes that can be made to a particular message or set of messages:

<b>Suppression</b>	Message appears in a hardcopy log but not on a console. SUP(YES/NO)
<b>Automation</b>	This lets the automation subsystem know to process a particular message. AUTO(YES/NO/token) Specifying AUTO(YES) will route the message to EMCS consoles with the AUTO attribute.
<b>Presentation</b>	This controls color, highlighting, and intensity attributes that the system uses when displaying messages on an operator console. You can specify these attributes on the .MSGCOLR statement.

### 9.2.3 MPF processing exit

MPF processing is specific to a certain type of message or a particular message ID that is defined in the MPFLSTxx parmlib member. Figure 9-1 on page 109 shows when MPF processing is invoked when a message is issued.

This exit is used primarily to do the following kinds of processing of a message:

- ▶ Modify the presentation of a message
- ▶ Modify the routing of a message
- ▶ Suppress or affect the automation of a message

#### Message flood use of this exit

Message Flood Automation is implemented as a message processing facility (MPF) IEAVMXIT routine that is called as a part of z/OS WTO processing. The exit examines each message in the z/OS system, and attempts to identify when too many WTOs are being issued and by whom.

It then takes appropriate action, usually to suppress the message from being displayed at a z/OS console, and to indicate that automation processing is not required. It can also issue commands to cancel the user or process.

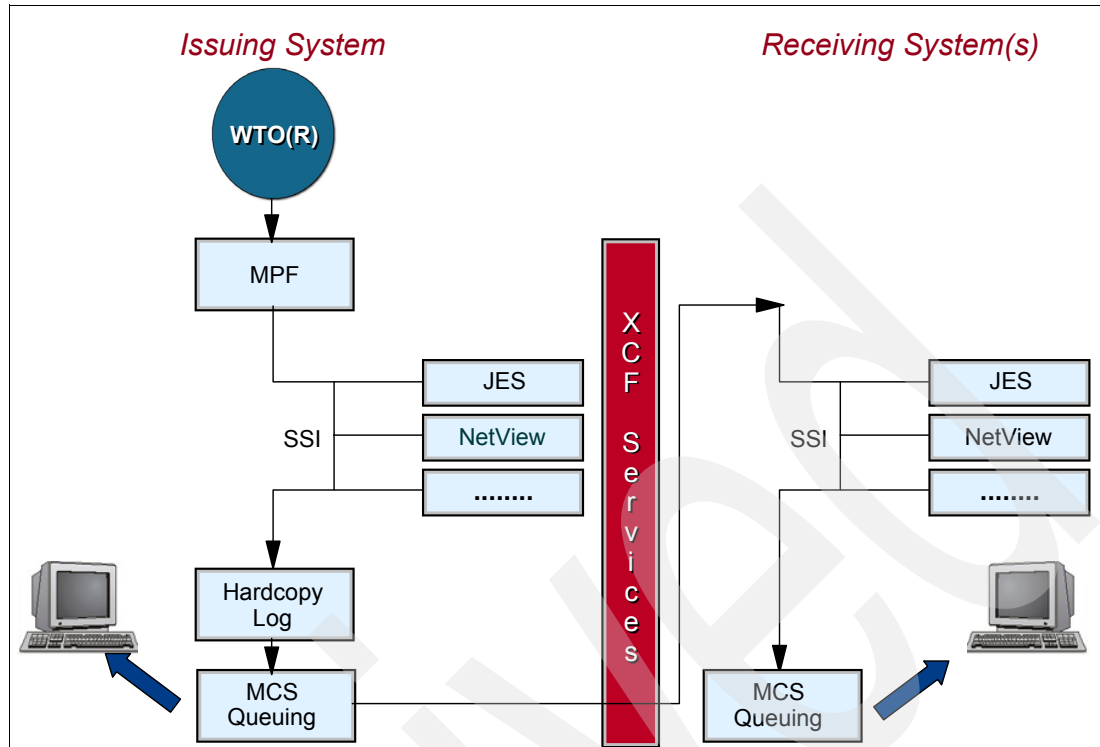


Figure 9-1 Message processing exit (MPF or IEAVMXIT)

## 9.3 Installing Message Flood Automation

Message Flood Automation does not ship a part named IEAVMXIT because this might cause the Message Flood Automation function to be inadvertently installed and activated when it is not desired to use this function. Message Flood Automation consists of the following load modules in SYS1.LINKLIB:

- ▶ CNZZCMXT
- ▶ CNZZVMXT

### 9.3.1 Message Flood Automation exits

Message Flood Automation uses two system exit points; IEAVXMIT as a general message exit, and CNZZCMXT as a system command exit. The following sample assembly language programs are in SYS1.SAMPLIB. You select one of the sample programs from SYS1.SAMPLIB, then assemble it and link it to the Message Flood Automation CNZZVMXT load module.

- ▶ CNZZVXT1
- ▶ CNZZVXT2

**Note:** Refer to *z/OS MVS Installation Exits*, SA22-7593, for additional information about the installation and use of the message and command exits. Install the IEAVMXIT in every SYS1.LINKLIB you use.

## IEAVMXIT migration considerations

Either CNZZVXT1 or CNZZVXT2 can be used if you do not already have an IEAVMXIT message exit. CNZZVXT1 is the simplest to use, but CNZZVXT2 can be used without change, if desired. The difference is that CNZZVXT2 has additional complexity that CNZZVXT1 does not.

- ▶ CNZZVXT1 provides a stub if there is no IEAVMXIT routine installed and Message Flood Automation is required; CNZZVXT1 will invoke the CNZZVMXT routine.
- ▶ CNZZVXT2 invokes the Message Flood Automation message exit CNZZVMXT using a slightly modified calling mechanism. CNZZVXT2 is documented to show how to place installation function in the exit both before and after the invocation of the Message Flood Automation message exit CNZZVMXT.

**Note:** In this implementation, CNZZVMXT returns to CNZZVXT2, not to the *caller* of CNZZVXT2. The CNZZVXT1 implementation has CNZZVMXT return to the caller of CNZZVXT1, *not* to CNZZVXT1.

If you already have an IEAVMXIT exit installed, you will need to do one of the following:

- ▶ Either put the invocation of Message Flood Automation into your IEAVMXIT using sample program CNZZVXT2 as an example of how to do this.
- ▶ Or fit your existing IEAVMXIT logic into the sample program CNZZVXT2 and rename it IEAVMXIT.

## MPFLSTxx parmlib member

At IPL, the system uses the MPFLSTxx member or members indicated on the MPF keyword on the INIT statement in CONSOLxx parmlib member. You can specify multiple MPFLSTxx members on the MPF keyword. In a sysplex, MPF processing has system scope; thus, you must plan MPF processing on each system in the sysplex.

The MPFLSTxx parmlib member contains statements that can affect the display, automation and retention of messages. During MPF processing, the RETAIN@, AUTO and SUP parameters on an MPFLSTxx parmlib member are processed first. Then either a user exit (specified by the USEREXIT parameter) is invoked or IEAVMXIT is invoked—but not both. A small part of a MPFLSTxx parmlib member is shown in Figure 9-2.

```
.NO_ENTRY,SUP(NO),RETAIN(I),AUTO(YES)
.DEFAULT,SUP(NO),RETAIN(NO),AUTO(NO)
IST1051I,SUP(YES),RETAIN(YES),AUTO(YES)
IST1062I,SUP(YES),RETAIN(YES),AUTO(YES)
AOF*,SUP(NO),RETAIN(NO),AUTO(YES)
CSA*,SUP(YES),RETAIN(NO),AUTO(YES)
EQQ*,SUP(NO),AUTO(YES)
EVJ*,SUP(NO),AUTO(YES)
IXG054A,USEREXIT(MPFOPLC)
IEF125I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEF403I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEF126I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEF404I,USEREXIT(MPFASID),RETAIN(NO),SUP(NO),AUTO(YES)
IEE391A,USEREXIT(MPFSMFC)
IEE366I,USEREXIT(MPFSMFC)
BDT3130,USEREXIT(MPFSBDTN)
```

Figure 9-2 MPFLSTxx parmlib member

## Using the IEAVMXIT exit

Message Flood Automation message processing runs as IEAVMXIT. Therefore, it can override the RETAIN, AUTO and SUP specifications set by the MPFLSTxx entry for the message or set by the NO\_ENTRY specification.

Figure 9-3 indicates the point at which the IEAVMXIT exit is entered during MPF processing. For the message IDs shown in Figure 9-2 on page 110, the ones with the USEREXIT parameter will use the specific MPF exit specified as shown in Figure 9-3. All other messages will use the IEAVMXIT exit, which goes through Message Flood Automation; this explains why individual messages do not go through both paths. Therefore, Message Flood Automation cannot override specifications set by individual MPF exits.

**Note:** If an MPFLSTxx entry does not exist for a message, then the settings from the NO\_ENTRY specification are applied. NO\_ENTRY allows you to specify the default processing you want for messages that are *not* identified in any of the active MPFLSTxx parmlib members.

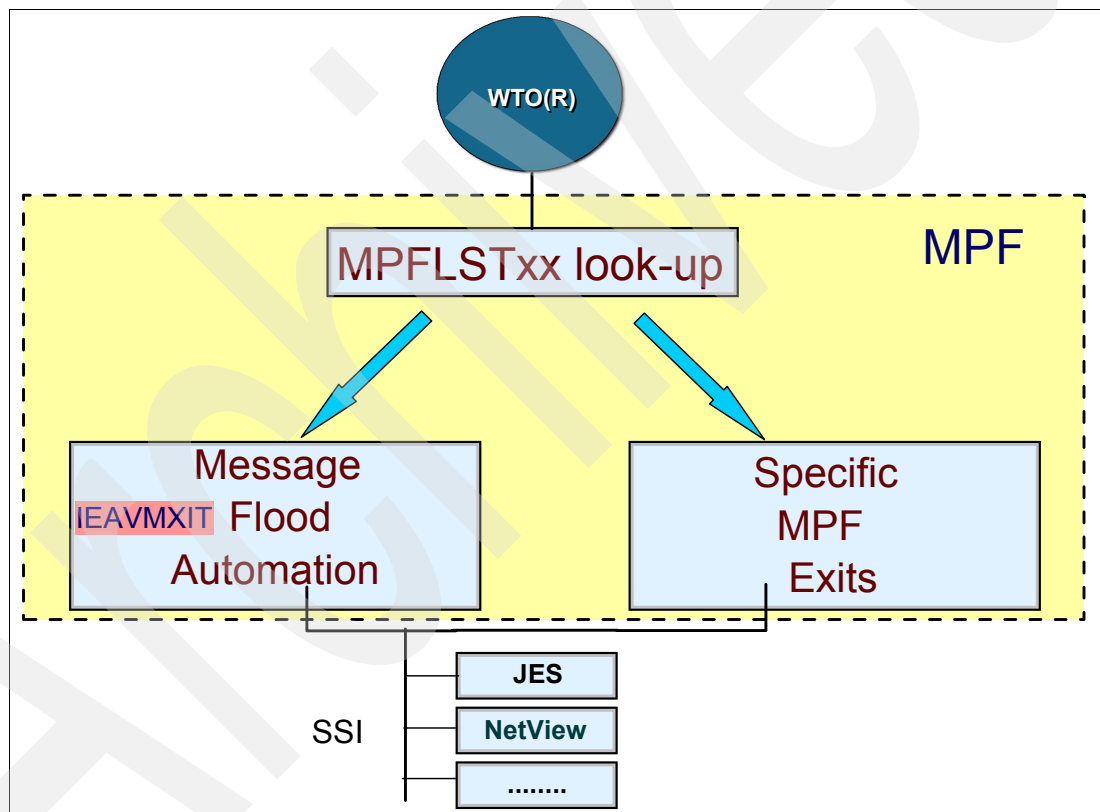


Figure 9-3 Message Flood Automation IEAVMXIT exit

**Note:** Use CNZZVXT2 and either integrate your existing IEAVMXIT function into it, or use CNZZVXT2 as an example of how to place the invocation of Message Flood Automation into your existing IEAVMXIT message exit. See *z/OS MVS Planning: Operations*, SA22-7601, for additional interface information.

The level of the message exit and the level of the command exit *must* be the same because they both map the same shared data area. There is code in both to ensure that they are at the

same level. This is primarily a concern for GDPS customers who are migrating from a GDPS level of Message Flood Automation to the z/OS level of Message Flood Automation.

The selection of the SYS1.SAMPLIB member and assembly should be done as follows:

1. Assemble it with the high-level assembler.
2. Link the assembled member to CNZZVMXT using the linkage editor or Binder.
3. Make changes in the CONSOLxx parmlib member and the MPFLSTxx parmlib members.

### Assemble sample code for the exit

Figure 9-4 shows sample JCL to assemble CNZZVXT1 and place it into an existing data set userid.SAMPLIB.OBJ.

```
//ASM EXEC PGM=ASMA90,REGION=6144K,  
// PARM=('OBJECT,NODECK,XREF(SHORT),LIST(133),ALIGN',  
// 'MACHINE(ZS-2,LIST),GOFF')  
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR  
// DD DSN=SYS1.MODGEN,DISP=SHR  
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(20,5)),DSN=&SYSUT1  
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(20,5)),DSN=&SYSUT2  
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(20,5)),DSN=&SYSUT3  
//SYSPRINT DD SYSOUT=A  
//SYSPUNCH DD DUMMY  
//SYSLIN DD DSN=userid.SAMPLIB.OBJ(CNZZVXT1),DISP=OLD  
//SYSIN DD DSN=SYS1.SAMPLIB(CNZZVXT1),DISP=SHR
```

Figure 9-4 JCL model to assemble CNZZVXT1

### Linking CNZZVXT1 or CNZZVXT2 with CNZZVMXT

Figure 9-5 shows sample JCL to link either CNZZVXT1 or CNZZVXT2 to CNZZVMXT. The result of the link will be a part named IEAVMXIT in SYS1.LINKLIB. For the SYSLMOD, you may use any data set in the LINKLIB concatenation.

CNZZVXT1, CNZZVXT2, CNZZVMXT and CNZZCMXT are all AMODE=31 and RMODE=ANY.

```
//LKED EXEC PGM=IEWL,REGION=0M,  
// PARM='XREF,LIST,RENT,REUS,AC(0)'  
//SYSPRINT DD SYSOUT=A  
//BASE DD DSN=SYS1.LINKLIB,DISP=SHR  
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))  
//SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR  
//SYSLIN DD DSN=userid.SAMPLIB.OBJ(CNZZVXT1),DISP=SHR  
// DD *  
SETCODE AC(0)  
INCLUDE BASE(CNZZVMXT)  
ENTRY IEAVMXIT  
NAME IEAVMXIT(R)
```

Figure 9-5 JCL model to link either CNZZVXT1 or CNZZVXT2 to CNZZVMXT

**Important:** The attribute RF was removed from members CNZZCMXT and CNZZVMXT in SYS1.LINKLIB in z/OS Version 1 Release 9. You *must* remove the parameter REFR from your linking JCL when you execute in a Version 1 Release 9 system.



## Creating an SMP/E ++USERMOD

As an alternative to manually assembling and linking CNZZVXT1 or CNZZVXT2 with CNZZVMXT, you can use SMP/E to perform the operation for you, creating an SMP/E ++USERMOD.

IBM recommends using the SMP/E ++USERMOD approach because that method allows SMP/E to automatically determine if and when your user exit must be relinked. It is necessary to keep both the user exit and the IBM load modules synchronized, and SMP/E is best positioned to do this automatically for you.

**Note:** As an example of SMP/E ++USERMOD, refer to “Creating an SMP/E ++USERMOD” in *z/OS MVS Planning: Operations*, SA22-7601.

## CONSOLxx and MPFLSTxx parmlib member considerations

Installing the Message Flood Automation code into the system libraries is not sufficient. You must also make changes to two SYS1.PARMLIB members:

► **MPFLSTxx parmlib member (using CNZZCMXT)**

Add a (dot) .CMD USEREXIT entry in a MPFLSTxx parmlib member. The .CMD entry causes the Message Flood Automation command exit to be automatically loaded at IPL or whenever a **SET MPF=** command is processed. For example:

```
.CMD USEREXIT(CMDRPL,CMDMVS,CMDGSYS,CMDCPF,CNZZCMXT)
```

Where CNZZCMXT is the Message Flood Automation command exit and the other exit names are installation-defined. If you already have one or more command exits specified, simply add CNZZCMXT to the existing specification.

Add CNZZCMXT either before or after any existing exits. The .CMD statement supports a maximum of 6 exit specifications.

► **CONSOLxx parmlib member (using IEAVMXIT)**

Add a UEXIT(Y) parameter to the INIT statement in the CONSOLxx parmlib member. UEXIT(Y) is the default and it is recommended that you explicitly code UEXIT(Y) in order to cause the automatic loading of the Message Flood Automation message exit at IPL and when **K M,UEXIT** command is processed. This is not strictly necessary because the default is UEXIT(Y), but explicitly specifying UEXIT(Y) can serve to remind you that a user exit is in use.

```
INIT      MLIM(1500)  
          RLIM(999)  
          LOGLIM(6000)  
          AMRF(Y)  
          APPLID(SCSMCS&SYSCLONE.)  
          UEXIT(Y)
```

You must make sure that the INIT statement in the CONSOLxx parmlib member being used does not have UEXIT(N) specified.

## 9.3.2 Loading and activating

The Message Flood Automation code can be loaded dynamically; there is no need to do an IPL. If you use the Link List Look aside facility, you will need to perform a Link List Lookaside refresh to bring the Message Flood Automation code into storage.

The CNZZCMXT command exit routine will be automatically loaded into storage after it has been link-edited into a data set in the LINKLIB concatenation and an LLA refresh (or IPL) has been performed.

- ▶ If you make the command exit available by changing the libraries referred to in the LINKLIST concatenation, you must issue the following command:

```
SETPROG LNKLST,UPDATE,JOB=*MASTER*
```

If you manipulate the libraries that are in your link list concatenation, refer to *z/OS MVS Planning: Operations, SA22-7601*, for details about using the **SETPROG** command.

After the Message Flood Automation code has been placed into storage, a **K M,UEXIT=Y** command can be used to actually load and enable the Message Flood Automation message exit. Likewise, a **SET MPF=** command can be used to load and activate the Message Flood Automation command exit.

Because the command exit uses data structures that are set up by the message exit, it is best to load and activate the command exit *after* loading and enabling the message exit. The reverse sequence will also work, but the command exit will be unable to perform any Message Flood Automation commands until the message exit has been loaded and creates the shared data structures.

Do the following in order to load and activate Message Flood Automation dynamically:

1. Use a **F LLA,REFRESH** command to load the Message Flood Automation code.
2. Use a **K M,UEXIT=Y** command to activate IEAVMXIT and CNZZVMXT.
3. Use a **SET MPF=** command to reload MPFLSTxx and cause the .CMD entry to be processed, loading the Message Flood Automation command exit.

```
F LLA,REFRESH
CSV210I LIBRARY LOOKASIDE REFRESHED
K M,UEXIT=Y
CNZZ016I Message Flood Automation policy initialized.
IEE712I CONTROL PROCESSING COMPLETE
SET MPF=J3
IEE252I MEMBER MPFLSTJ3 FOUND IN SYS1.PARMLIB
IEE712I SET MPF PROCESSING COMPLETE
D MF,STATUS
CNZZ042I Message Flood Automation V2ROM00 DISABLED. 289
Policy INITIALIZED. Using PARMLIB member: internal
Intensive modes: REGULAR-OFF ACTION-OFF SPECIFIC-OFF
Message rate monitoring DISABLED. 0 msgs 0 secs
```

Figure 9-6 Expected messages to load and enable Message Flood Automation

**Note:** These commands load and enable Message Flood Automation message exit (IEAVMXIT) and command exit (CNZZCMXT) with the internal policy. After this, you must load your policy by using the **SET MSGFLD=** command, and turn Message Flood Automation on by issuing a **SETMF ON** command.

## 9.4 Customization and tuning

An installation policy to control message flooding situations is specified through the new MSGFLDxx parmlib member.

Message Flood Automation is performed on all subsequent messages using either default actions or actions specified through a MSGFLDxx parmlib member after a **SETMF ON** command is issued that activates Message Flood Automation.

### 9.4.1 Providing a MSGFLDxx parmlib member

Prepare one or more MSGFLDxx members of PARMLIB and customize the message threshold values provided before enabling Message Flood Automation on your system.

You may have as many MSGFLDxx parmlib members as you want, but Message Flood Automation only supports one member being active at a time. Message Flood Automation processing requires that the MSGFLDxx suffix (the xx) be alphabetic or numeric. National or other special characters are *not* supported.

IBM provides a usable sample MSGFLDxx parmlib member named CNZZMFxx in the SYS1.SAMPLIB library. However, it is recommended that you customize this sample MSGFLDxx member before you attempt to use it.

#### MSGFLDxx statements

The following statement types are provided in MSGFLDxx parmlib member:

- ▶ Comment statements
- ▶ Msgtype statements  
REGULAR, ACTION, and SPECIFIC
- ▶ DEFAULT statements  
The DEFAULT statement specifies the default action to be taken for a specific address space that exceeds the job threshold message rate, or a specific message that exceeds the message threshold message rate.
- ▶ DEFAULTCMD statements  
The DEFAULTCMD statement specifies the default command that will be issued if a CMD action has been specified for the address space.
- ▶ JOB statements  
JOB statements identify up to 10 specific jobs for which specific actions are to be taken if REGULAR or ACTION messages from the job are involved in a message flood.
- ▶ MSG statements  
The MSG statement defines up to 30 specific messages for which specific actions are to be taken if the message is involved in a message flooding situation.

Figure 9-7 on page 116 displays the sample MSGFLDxx parmlib member CNZZMFXX that is supplied in SYS1.SAMPLIB.

```

/*-----*/
/* Sample MSGFLDxx PARMLIB member. */
/*-----*/
/*-----*/
/* REGULAR message specifications. */
/*-----*/
REGULAR MSGTHRESH=50,JOBTHRESH=20,INTVLTIME=1
REGULAR SYSIMTIME=2,JOBIMTIME=2
DEFAULT LOG,NOAUTO,NODISPLAY,NOCMD
DEFAULTCMD '&,CANCEL & -- cancelled by Message Flood Automation'
JOB AOC%NV* AUTO
JOB LLA* AUTO
JOB ZAP1 CMD
/*-----*/
/* ACTION message specifications. */
/*-----*/
ACTION MSGTHRESH=50,JOBTHRESH=20,INTVLTIME=1
ACTION SYSIMTIME=2,JOBIMTIME=2
DEFAULT LOG,NOAUTO,NODISPLAY,NOCMD,NORETAIN
DEFAULTCMD '&,CANCEL & -- cancelled by Message Flood Automation'
JOB AOC%NV* AUTO,RETAIN
JOB LLA* AUTO
JOB ZAP2 CMD
/*-----*/
/* SPECIFIC message specifications. */
/*-----*/
SPECIFIC MSGTHRESH=50,INTVLTIME=1
SPECIFIC SYSIMTIME=2
SPECIFIC MSGIMTIME=2
SPECIFIC MSGLIMIT=20
DEFAULT LOG,NOAUTO,NODISPLAY,NORETAIN
MSG IOS001E
MSG IOS003A
MSG IOS050I
MSG IOS051I
MSG IOS071I
MSG IOS251I
MSG IOS444I
MSG IOS450E

```

Figure 9-7 Sample MSGFLDxx parmlib member

**Note:** Refer to *z/OS MVS Planning: Operations, SA22-7601*, for more detailed information about the parameters.

## 9.4.2 Types of message classes processed

The Message Flood Automation processing handles three separate types of message classes. The definitions of how these classes are defined is in the MSGFLDxx parmlib member. Each class of messages is handled separately.

Message Flood Automation can take action against “privileged” messages which are queued to consoles even in storage shortage situations. The message classes are as follows:

- ▶ SPECIFIC messages

- A set of messages identified by the installation that are to be handled separately

- ▶ ACTION messages
 

Messages that have one or more of the following descriptor codes set:

  - 1 - System failure messages (typically messages with a W (wait) message ID suffix)
  - 2 - Immediate action required messages (typically messages with an A (action) or D (decision) message ID suffix)
  - 3 - Eventual action required messages (typically messages with an E (eventual action) message ID suffix)
  - 11 - Critical eventual action required messages (typically messages with an E (critical eventual action) message ID suffix)
- ▶ REGULAR messages
 

Messages that do not fall into any of the preceding categories.

**Note:** REGULAR messages include the “command echo” messages which put the text of a command into the display area of a console and place the text of a command into the SYSLOG and OPERLOG. Descriptor codes describe the significance of messages.

They indicate whether the system or a task stops processing, waits until some action is completed, or continues. This code also determines how the system will display and delete the message.

### 9.4.3 Message class controls

Each class has its own set of controls, as shown in Figure 9-8. Message Flood Automation classes run in one of two modes: normal or intensive. Each set of controls operate independently; for example, the system may be in intensive mode for regular messages but not for action messages. The effect is that action messages will still be processed by z/OS in the normal way.

```

INTVLTIME (Interval limit time)
JOBIMTIME (Job inter-message mean time) - not supported for SPECIFIC
JOBTHRESH (job message threshold) - not supported for SPECIFIC
MSGIMTIME (Message inter-message time) - not supported for REGULAR and ACTION
MSGLIMIT (Message threshold limit) - not supported for REGULAR and ACTION
MSGTHRESH (Message threshold)
SYSIMTIME (System inter-message time)
  
```

Figure 9-8 Message class controls that are defined in parmlib

The use of these message class controls for each message class is shown in Figure 9-9 on page 121.

#### Message class modes

The Message Flood Automation normal and intensive class modes are explained here.

**Normal mode** In normal mode, messages are counted. When a threshold number (MSGTHRESH) of messages has been counted, the time taken to count those messages is determined. If the time is less than a limit value (INTVLTIME), the system is placed into intensive mode. It is expected that this determination is likely to be done relatively infrequently, every 50 to 100 messages or more. The INTVLTIME value should be set to identify high message rates. A value of 5 seconds for INTVLTIME

indicates an average rate of 20 messages per second if MSGTHRESH is set to 100.

The processing overhead in normal mode is therefore very small. Only a very small number of instructions are executed in the exit for each message. No dynamic storage is obtained or freed, and no recovery environment is established.

### **Intensive mode**

In intensive mode, each message is subject to extra processing. Messages are counted for each address space (up to a maximum of 10) issuing messages and compared to a further limit value (JOBTHRESH). If any one address space issues JOBTHRESH messages within INTVLTIME, then it is subject to action from that time on. This action may be installation-specified, but is typically defaulted to be no-display and no-automation.

At the end of each interval of MSGTHRESH messages, a check is made to see if intensive mode should be maintained, and whether address spaces in "act-upon mode" should remain so. Message bursts can end suddenly. The address space that issues them may suddenly exit a tightloop condition and resume normal processing. In this circumstance, it is likely that subsequent messages are important and should be processed normally. To allow this to happen, there are two further controls: system inter-message time (SYSIMTIME), and job (or message) inter-message time (JOBIMTIME or MSGIMTIME).

When in intensive mode, if the time since the last message is greater than SYSIMTIME, then intensive mode is discontinued. This ensures that the first message after a break is not acted upon.

Similarly, if an address space is in act-upon mode, and the time since its last message exceeds the JOBIMTIME, then it is removed from act-upon mode. For specific messages, if a message is in act-upon mode, and the time since the last message exceeds the MSGIMTIME, then the message is removed from act-upon mode.

### **Use of message class controls**

Table 9-1 on page 119 shows the combinations of message class control keywords that are supported for each message class. As mentioned, each class of messages is handled separately, and has its own set of controls (MSGTHRESH, INTVLTIME, and so on). Each set of controls operates independently; the system can be in intensive mode for regular messages, but not for action messages. z/OS still processes action messages in the normal way.

The control algorithms for regular and action messages are identical and are as described previously. For specific messages, the control algorithm is similar although it is applied to individual messages and not to jobs or address spaces.

The MSGLIMIT parameter performs the same function in specific message processing that the JOBTHRESH parameter performs in regular and action message processing. The MSGIMTIME parameter performs the same function in specific message processing that the JOBIMTIME parameter performs in regular and action message processing, although it is applied against specific messages rather than address spaces.

Table 9-1 Message class type and keyword combinations

msgtype=regular	msgtype=action	msgtype=specific
INTVLTIME	INTVLTIME	INTVLTIME
JOBIMTIME	JOBIMTIME	
JOBTHRESH	JOBTHRESH	
		MSGIMTIME
		MSGLIMIT
MSGTHRESH	MSGTHRESH	MSGTHRESH
SYSIMTIME	SYSIMTIME	SYSIMTIME

### Message class control default values

The system default values, either before the MSGFLDxx parmlib member is read, or accepting the IBM defaults, are as follows:

```
REGULAR MSGTHRESH=50
REGULAR JOBTHRESH=20
REGULAR INTVLTIME=1
REGULAR SYSIMTIME=2
REGULAR JOBIMTIME=2
```

```
ACTION MSGTHRESH=50
ACTION JOBTHRESH=20
ACTION INTVLTIME=1
ACTION SYSIMTIME=2
ACTION JOBIMTIME=2
```

```
SPECIFIC MSGTHRESH=50
SPECIFIC MSGLIMIT=20
SPECIFIC INTVLTIME=1
SPECIFIC SYSIMTIME=2
SPECIFIC MSGIMTIME=2
```

### Message class control value ranges

The value for JOBTHRESH, MSGLIMIT, and MSGTHRESH is a positive, non-zero integer count of messages in the range 1 to 999999999.

The value for INTVLTIME is a positive, non-zero integer time in seconds in the range 1 to 999999999.

The value for SYSIMTIME, JOBIMTIME and MSGIMTIME is a positive, non-zero floating point time in seconds in the range 0.000001 to 16777215.0.

### Operator commands to display individual Message Flood Automation

The values specified for each of the message class controls can be displayed by the following operator command. The command can be used in any of the following forms:

```
DISPLAY MSGFLD,MSGTYPE=msgtype,keyword
DISPLAY MF,MSGTYPE=msgtype,keyword
D MSGFLD,MSGTYPE=msgtype,keyword
D MF,MSGTYPE=msgtype,keyword
```

For example:

```
D MF,MSGTYPE=REGULAR,MSGTHRESH
CNZZ301I Value of REGULAR MSGTHRESH is 20
```

#### 9.4.4 Message Flood Automation guidelines

Message Flood Automation thresholds should be set based on the mean (most common) message rate, not on the maximum message rates. The following guidelines should be used when defining the message class controls:

- ▶ The REGULAR message threshold (MSGTHRESH) should be set somewhat higher than the mean message rate, and your REGULAR message inter-message time (SYSIMTIME) at or slightly below the mean message rate inter-message time.
- ▶ The ACTION job message threshold (JOBTHRESH) must be set to a value less than that of MSGTHRESH. A JOBTHRESH value that is 30 to 40% of MSGTHRESH is a useful starting point.
- ▶ The SPECIFIC MSG message threshold (MSGLIMIT) must be set to a value less than that of MSGTHRESH. A MSGLIMIT value that is 15 to 20% of MSGTHRESH is a useful starting point.

##### Guideline examples

Setting MSGTHRESH=50 and INTVLTIME=1 specifies a message rate of 50 messages/second. Setting MSGTHRESH=100 and INTVLTIME=2 also specifies a message rate of 50 messages/second. You can use different combinations of threshold and interval to trade off message flood detection responsiveness and message flood detection overhead.

The general idea is to set the various thresholds high enough that they are not being triggered by normal fluctuations in message rates but are triggered when sudden, very high message rates are encountered. For REGULAR messages, using one of the suggested threshold values provided by the CNZZ043I message is a good first approximation. Set your thresholds high enough that Message Flood Automation is not constantly oscillating into and out of intensive mode.

Using the MSGTHRESH=50 and INTVLTIME=1 specification makes Message Flood Automation more responsive to detecting message flooding situations because only 50 messages are counted between computations of the message rate. However, the overhead of the message rate computation is incurred twice as frequently as the MSGTHRESH=100 and INTVLTIME=2 specification.

##### Message Flood Automation message rate monitoring

It is very important that the message rate thresholds be properly set in the Message Flood Automation policy. You can determine the values that are appropriate for your system by using the Message Flood Automation Message Rate Monitoring function, which can be run without enabling any other Message Flood Automation function.

Run the Message Rate Monitoring function on the system that you intend to run Message Flood Automation on, and not on a test system. It is also very important that you obtain a representative sample. We recommend that you run the Message Rate Monitoring function for a 24-hour period that encompasses some of your busiest processing time.





## Tuning MSGFLDxx parmlib member

Set the REGULAR message MSGTHRESH value from the “suggested threshold” values provided at the bottom of the Message Rate Monitoring graph. We recommend using the 99% threshold value. This means that in the sample, 99% of the messages occurred at a message rate lower than the threshold value. Use of this threshold value will ensure that Message Flood Automation is not taking action unnecessarily.

The interval time value should be set to one for all of the message types. Setting the value to one will allow Message Flood Automation to be responsive without entailing undue overhead.

For the REGULAR and ACTION message types, the job threshold must always be *less* than the overall message threshold. These relationships are checked and will cause warning messages to be produced when the policy is loaded, or when the various values are displayed by operator command. A useful value for the job threshold is 20 to 30% of the respective overall message threshold.

For the SPECIFIC message type, the message limit must be less than the overall message threshold. This relationship is checked and will cause a warning message to be produced when the policy is loaded, or when the various values are displayed by operator command. A useful value for the message threshold is 10 to 15% of the respective overall message threshold.

The ACTION and SPECIFIC MSGTHRESH values should be set *lower* than the REGULAR MSGTHRESH value because the messages in these classifications typically occur much less frequently than more common messages.

## 9.4.5 Turning Message Flood Automation ON or OFF

Message Flood Automation command processing becomes active as soon as the message exit and command exits are loaded and enabled. This will occur automatically during an IPL and whenever a **SET MPF=** command is processed that has a (dot) .CMD entry for the Message Flood Automation command exit.

Message Flood Automation message processing will be loaded and enabled automatically at IPL and whenever a **K M, UEXIT(Y)** command is processed. Although Message Flood Automation message processing is automatically loaded and enabled, no message processing will occur until it is explicitly turned on by operator command.

Note that Message Flood Automation can be affected by **SET MPF=xx** commands that have nothing to do with Message Flood Automation processing. For example, if a **SET MPF=** command loads an MPFLSTxx parmlib member that does not have the .CMD statement for loading the command exit and Message Flood Automation was previously active, Message Flood Automation command processing will be deactivated. (The symptom for this is that none of the Message Flood Automation commands will be recognized.)

After the Message Flood Automation code has been installed in the system libraries and the various parmlib members have been customized, you can use the following commands to turn Message Flood Automation On or OFF:

- ▶ Load your Message Flood Automation policy

```
SET MSGFLD=xx
```

or

```
T MSGFLD=xx
```

where xx is the suffix of a MSGFLDxx parmlib member.

- ▶ Turn Message Flood Automation message processing ON  
SETMF ON
- ▶ Turn Message Flood Automation message processing OFF  
SETMF OFF

```

SET MSGFLD=00
CNZZ016I Message Flood Automation policy initialized.
CNZZ401I Message Flood Automation loading: MSGFLD00
CNZZ410I Message Flood Automation loading of MSGFLD00 complete.
SETMF ON
CNZZ041I Message Flood Automation ENABLED.    PARMLIB member:MSGFLD00
D MF,STATUS
CNZZ042I Message Flood Automation V2ROM00  ENABLED.
Policy INITIALIZED.    Using PARMLIB member: MSGFLD00
Intensive modes: REGULAR-OFF ACTION-OFF SPECIFIC-OFF
Message rate monitoring DISABLED.           0 msgs           0 secs
SETMF OFF
CNZZ041I Message Flood Automation DISABLED.  PARMLIB member:MSGFLD00
D MF,STATUS
CNZZ042I Message Flood Automation V2ROM00  DISABLED.
Policy INITIALIZED.    Using PARMLIB member: MSGFLD00
Intensive modes: REGULAR-OFF ACTION-OFF SPECIFIC-OFF
Message rate monitoring DISABLED.           0 msgs           0 secs

```

Figure 9-10 Expected messages when loading and enabling Message Flood Automation

**Note:** You cannot use a COMMNDxx parmlib member to issue **SET MSGFLD** commands to load the MSGFLDxx parmlib member because COMMNDxx processing occurs prior to the availability of the system services required to read SYS1.PARMLIB.

Also, you cannot use COMMNDxx to issue the **SETMF ON** command to enable Message Flood Automation because COMMNDxx processing occurs before the CNZZCMXT command exit is automatically loaded by the system during IPL.

## 9.4.6 Displaying your policy

Almost all of the currently active Message Flood Automation policy can be displayed using the **DISPLAY MSGFLD,parameters** command.

Almost all of the Message Flood Automation policy can be modified, when necessary, by the operator using the **SETMF** command. Consult *z/OS MVS Planning: Operations, SA22-7601*, for details of the **SETMF** command syntax. Because changes made by the **SETMF** command will persist only until the next IPL, IBM recommends that changes that you want to be permanent be made in your MSGFLDxx parmlib member. You may find the **SETMF** command useful during Message Flood Automation testing and during emergency situations.

- ▶ Display your Message Flood Automation policy, as follows:
  - DISPLAY MSGFLD,PARAMETERS

The **PARAMETERS** option displays in tabular form all of the threshold values and all of the time periods over which they are evaluated.

- DISPLAY MSGFLD,DEFAULTS

The DEFAULTS option will display in tabular form all of the default actions that will be taken for each of the three message classes: SPECIFIC messages, ACTION messages and REGULAR messages.

- DISPLAY MSGFLD,JOBS

The JOBS option will display in tabular form the names of all of the jobs that will have unique actions taken for them.

- DISPLAY MSGFLD,MSGS

The MSGS option will display in tabular form the identifiers of all of the messages that will have unique actions taken for them.

- ▶ Dynamically modify your Message Flood Automation policy with the following command:
  - SETMF command

## 9.5 Command summary

Operator commands exist to do the following functions for Message Flood Automation:

- ▶ Enable message flood checking
- ▶ Disable message flood checking
- ▶ Reinitialize the counts, indicators and actions, and read the specified MSGFLDxx parmlib member
- ▶ Display the status of the Message Flood Automation function
- ▶ Display whether intensive mode is active for the different classes of messages
- ▶ Display the counts and parameters used by Message Flood Automation
- ▶ Modify the counts and parameters used by Message Flood Automation
- ▶ Enable message rate information gathering
- ▶ Disable message rate information gathering
- ▶ Display message rate information
- ▶ Free the common storage area that is used by Message Flood Automation

All of these commands are implemented through a formal z/OS command exit (CNZZCMXT).

Message Flood Automation **SET**, **SETMF**, and **DISPLAY** commands do not perform authorization checks and therefore cannot be restricted through the installation's security product.

You can use the following to perform **DISPLAY**, **SET**, and **SETMF** Message Flood Automation commands.

**Note:** Refer to *z/OS MVS Planning: Operations, SA22-7601*, for more detailed information about the commands.

### **DISPLAY MSGFLD command**

Figure 9-11 on page 125 displays commands for all of the Message Flood Automation parameters.

```

DISPLAY MSGFLD,PARAMETERS
DISPLAY MSGFLD,DEFAULTS
DISPLAY MSGFLD,JOBS
DISPLAY MSGFLD,MSGS
DISPLAY MSGFLD,MODE
DISPLAY MSGFLD,MSGRATE[,n]
DISPLAY MSGFLD,STATUS

DISPLAY MSGFLD,MSGTYPE=ACTION{,JOBTHRESH}
                                {,MSGTHRESH}
                                {,INTVLTIME}
                                {,JOBIMTIME}
                                {,SYSIMTIME}

DISPLAY MSGFLD,MSGTYPE=REGULAR{,JOBTHRESH}
                                {,MSGTHRESH}
                                {,INTVLTIME}
                                {,JOBIMTIME}
                                {,SYSIMTIME}

DISPLAY MSGFLD,MSGTYPE=SPECIFIC{,MSGTHRESH}
                                {,INTVLTIME}
                                {,SYSIMTIME}
                                {,MSGIMTIME}
                                {,MSGLIMIT}

```

DISPLAY may be abbreviated D and MSGFLD may be abbreviated MF.

Figure 9-11 DISPLAY MSGFLD command

## SET MSGFLD command

Use this command to change to a new policy.

```

SET MSGFLD=xx

SET may be abbreviated T.

```

Figure 9-12 SET MSGFLD command

## SETMF commands

To modify the Message Flood Automation parameters being used, use the **SETMF** command. The **SETMF** commands, shown in Figure 9-13 on page 126, use the same msgtype and keyword specifications as the **DISPLAY MSGFLD** command, so any value that may be displayed by **DISPLAY MSGFLD** may be set using the **SETMF** command.

One or more keyword=value pairs may be specified, separated by a comma.

```

SETMF ON
SETMF OFF
SETMF FREE
SETMF MONITORON
SETMF MONITOROFF

SETMF MSGTYPE=ACTION{,JOBTHRESH=value}
                    {,MSGTHRESH=value}
                    {,INTVLTIME=value}
                    {,JOBIMTIME=value}
                    {,SYSIMTIME=value}

where value is a count of messages, or time in seconds
or fractions of a second (SYSIMTIME and JOBIMTIME only)

SETMF MSGTYPE=ACTION,
    DEFAULT=action[,action]

SETMF MSGTYPE=ACTION,JOB=jobname,
    [,action][,action]

where action is LOG|NOLOG, DISPLAY|NODISPLAY,
              AUTO|NOAUTO, RETAIN|NORETAIN,
              CMD|NOCMD

SETMF MSGTYPE=REGULAR{,JOBTHRESH=value}
                    {,MSGTHRESH=value}
                    {,INTVLTIME=value}
                    {,JOBIMTIME=value}
                    {,SYSIMTIME=value}

where value is a count of messages, or time in seconds
or fractions of a second (SYSIMTIME and JOBIMTIME only)

SETMF MSGTYPE=REGULAR,
    DEFAULT=action[,action]

SETMF MSGTYPE=REGULAR,JOB=jobname
    [,action][,action]

SETMF MSGTYPE=SPECIFIC{,MSGTHRESH=value}
                    {,INTVLTIME=value}
                    {,SYSIMTIME=value}
                    {,MSGIMTIME=value}
                    {,MSGLIMIT=value}

where value is a count of messages, or time in seconds
or fractions of a second (SYSIMTIME and MSGIMTIME only)

SETMF MSGTYPE=SPECIFIC,
    DEFAULT=action[,action]

SETMF MSGTYPE=SPECIFIC,MSG=msgid
    [,action][,action]

```

Figure 9-13 SETMF commands

## WLM enhancements

The z/OS Workload Manager (WLM) is enhanced with improved performance routing, priority settings, and cancel functionality, further improving on the mainframe's leadership position in workload management capabilities. With z/OS WLM, you can define business and performance goals customized for your applications. The z/OS system decides how much resource, such as CPU and storage, should be given to applications that serve the workload to meet the goal. WLM constantly monitors the system and adapts resource applications to meet application goals, taking into account not only server resources, but network traffic, router bottlenecks, application health, and transaction prioritization as well, thus providing autonomic, policy-based z/OS performance management that can be tuned to meet your applications' needs.

This chapter provides information about the WLM enhancements introduced in z/OS Version 1 Release 9. The following improvements are discussed:

- ▶ Promote jobs which have been cancelled
- ▶ Start minimum number of servers
- ▶ WLM/SRM enhancements for blocked workloads
- ▶ RMF enhancements for blocked workloads
- ▶ zAAPs and zIIPs support stage 3 routing

## 10.1 Promote jobs which have been cancelled

Occasionally you may need to cancel an address space because it is holding up other work, and you need to get that address space out of the system quickly. Sometimes, however, cancel processing takes a long time to complete because the job is being pre-empted by higher priority work and hence is not dispatched. When an address space is cancelled, the majority of cancel processing runs in the address space being cancelled, so it is running at the dispatch priority of the address space. The process of canceling this address space may take a long time to terminate because all processors are too busy processing work with higher importance.

### 10.1.1 z/OS V1R9 enhancement

WLM is now changed to increase the priority of canceled address spaces. SRM can now promote the address space being cancelled to a higher dispatch priority in order to give the address space sufficient access to a CP in order to have it terminate faster. This can eliminate the need to reset the priority of a canceled job, task, or user to speed address space termination when resolving resource contention issues.

### 10.1.2 Migration and coexistence considerations

The BRINGIN SYSEVENT, an internal service used during cancel processing, is no longer used by the z/OS operating system and is removed with z/OS V1R9. While the BRINGIN SYSEVENT was not a public interface, use of this function by other than the operating system can cause problems in production environments. If the current job in an address space has been canceled and if the BRINGIN service were not issued, an address space that had been swapped out because of a shortage might be kept out until the shortage had been relieved.

Use of this SYSEVENT with z/OS V1R9 causes an ABEND of the caller, abend code x'15F', reason code 4.

**Note:** Refer to *z/OS MVS System Codes*, SA22-7626, for more information about the abend code x'15F' and reason code 4.

A new CANCEL SYSEVENT is now used to request a swap in and promote of an address space. This allows for operator or user CANCEL processing to be done more quickly and release system resources. This new service is used to accelerate the cancel process.

When the CANCEL command for an address space has been accepted by the command processor, the command processor notifies SRM that cancel processing is starting for the address space.

**Note:** System events (SYSEVENTs) are indicated by an entry to system resources manager (SRM) through direct branch or SVC 95 (SVC X'5F'). SYSEVENTs appear in the generalized trace facility (GTF) and system trace records.



## 10.2 Start a minimum number of servers

Workload management creates as many server address spaces as are needed to meet the goals of the work running in the servers, unless the application has limited the number of server instances that workload management can create using IWM4SLI.

WLM starts and stops server address spaces depending on the workload and the system capabilities. This self-optimization logic is enhanced by an additional parameter that allows the application to decide in which way the minimum amount of servers are to be started by WLM.

**Note:** The IWM4SLI service was introduced with z/OS V1R6 to support the 64-bit address space.

### 10.2.1 z/OS V1R9 enhancement

The IWM4SLI service should be used to tell WLM the total number of server instances which are supported by the application. WLM will ensure that no more server instances will be started in the system.

Use the AE\_SERVERMAX parameter to establish a maximum number, which is particularly useful for applications such as MQSeries® and workflow that connects to back-end applications supporting a limited number of parallel connections. Use the AE\_SERVERMIN parameter to establish a minimum number, because this allows an application to keep a number of servers active, even during low utilization periods. In addition, you can specify AE\_SPREADADMIN=YES to ensure that the defined minimum number of servers are distributed evenly across all of the service classes used to execute work requests in the application environment.

#### New parameters

The new parameters that can be specified using the IWM4SLI service are:

- ▶ START\_MINIMUM=SERIAL
- ▶ START\_MINIMUM=PARALLEL

When AE\_SERVERMIN=ae\_servermin is specified it indicates whether WLM will start the minimum number of servers one by one or in parallel. The default is START\_MINIMUM=SERIAL.

#### **START\_MINIMUM=SERIAL**

The server tasks specified in AE\_SERVERMIN will be started one by one. This means the next server will only be started if the previous server has connected to WLM.

#### **START\_MINIMUM=PARALLEL**

The server tasks specified in AE\_SERVERMIN will be started in parallel. This means WLM will start additional servers even when the previous servers have not connected to WLM.

In such environments, the startup of the minimum servers can be accelerated. This new service used by WLM started server address spaces is called by the first server address space, which then decides in which way the remaining servers will be started.

## Server region support

When the first request is queued to an application environment, workload management detects that there are no active servers for the request, and automatically starts one. The MVS procedure name and start parameters are taken from the application environment definition in the service definition. As the workload fluctuates, workload management adjusts the number of server address spaces so the goals of the work are met.

When the server initializes, it must establish itself as a server address space using the IWM4CON service with SERVER\_MANAGER=YES parameter, and indicate which application environment it is servicing. The subsystem type and name specified on the server connect must match the values specified on the associated queueing manager connect. Immediately after invoking the IWM4CON service, the server region optionally establishes a maximum and/or minimum number of server instances that can be started for a given application environment.

The first server region that connects decides the minimum amount of server address spaces and also decides in which way they will be started (serial or parallel), as shown in Figure 10-1. If any server uses this service to define limits, the limits apply for *all* servers of the application environment, regardless of whether or not other servers use the service.

If a server defines new limits during execution, WLM attempts to meet the new limit definitions as soon as possible. If the maximum limit for servers is reduced during execution, it is not predictable when WLM is able to meet the new maximum definition. This depends highly on the execution time of the running work requests. Therefore, changing the limits during execution should be done very carefully and primarily during times of low application utilization.

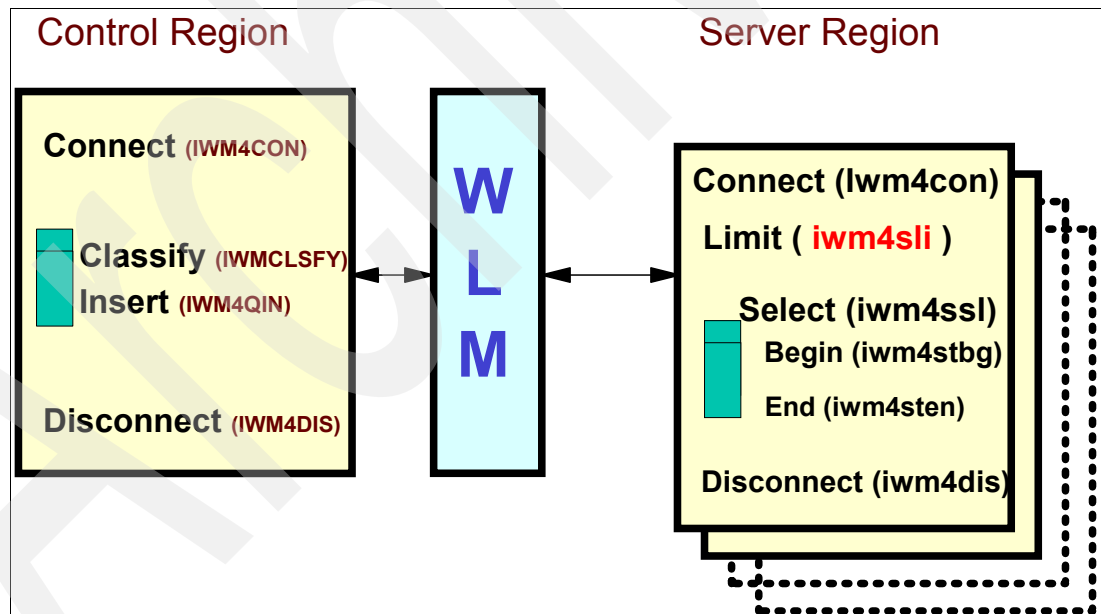


Figure 10-1 Server regions connecting to WLM

The logic that decides how many servers can be in which states is changed and has been extended by the special case that the minimum amount of servers is not reached. The new parameter on IWM4SLI decides if the servers need to connect before the next server start will be initiated by WLM, as shown in Figure 10-2 on page 131.

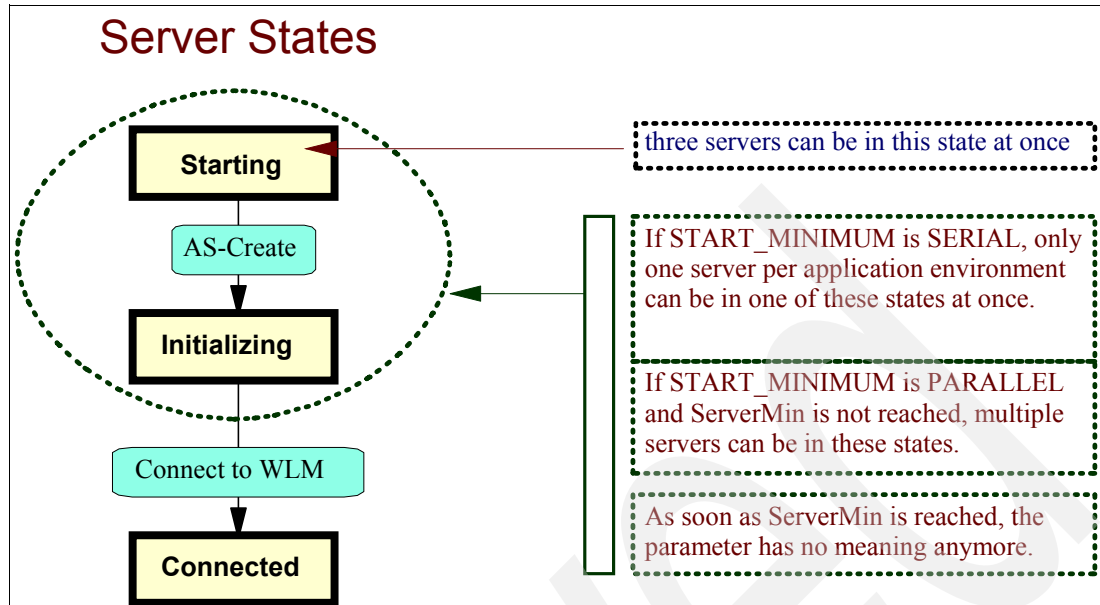


Figure 10-2 Server regions establishing a minimum or maximum number

The syntax of the WLM service IWM4SLI is described in *z/OS MVS Programming: Workload Management Services*, SA22-7619.

## 10.2.2 Exploiters of the new service request

Every application that uses the WLM services to manage its server address spaces and has a minimum limit for the amount of servers can use this new parameter.

The usage of this new parameter makes only sense if the application suffers from a long startup time of the minimum amount of servers specified on the IWM4SLI service.

If the long startup time is caused by a long processing time of the servers before they connect to WLM, this new parameter may speed up the total time needed to start up the minimum amount of servers.

## 10.3 WLM enhancements for blocked workloads

During periods of 100% CP utilization, it is possible that discretionary workloads (workloads defined by your installation to have lower dispatch priority) are not dispatched for execution. These discretionary workloads may obtain and hold serially reusable resources required by other workloads, which may block the progress of higher dispatch priority workloads. In z/OS V1R9, it is possible to specify that any address space and any enclave that has work that is ready to run but does not get CP service within a certain time interval can be temporarily promoted to a higher dispatching priority. Resource measurement facility (RMF) supports this function by reporting relevant measurements.

When low priority work obtains a resource and then gets suspended in favor of higher importance work, the resource that the high importance work needs is then blocked. The high priority work is now in effect blocked by the lower priority work and WLM resolves this with z/OS V1R9 by granting limited CP access to work units which could not get hold of a CP for an extended period of time. WLM periodically examines the IN-queue and identifies work

units which have been CP denied. The dispatching priority of these work units is *temporarily* raised to allow execution of a small number of instructions. This assumes that such short periods of CP access does not harm high importance work and could help low importance work to release locks and other critical resources.

### 10.3.1 Promote higher dispatch priority

WLM defined goals allow high importance work better access to system resources than lower importance work, but these goals cannot guarantee that all work will always get access to the system resources within a certain period of time. Given the high utilization of some systems, WLM is not able to ensure that all work can actually run in the system. In many cases this is not a problem, and installations are aware of the fact that lower importance work has to wait until resources become available. However, in certain cases, this can lead to unwanted consequences.

Because there are cases where WLM does not know what work is holding important resources, WLM gives work that did not get access to a CP for some elongated time period a small amount of access to the CPU. The z/OS V1R9 support periodically examines the IN-queue and identifies work units which have been CP-starved for an extended period of time. WLM can then promote these work units and guarantee that the promoted work gets access to a CP, but cannot take over the CP.

This can eliminate the need to manually increase the priority of low priority work holding a resource to accelerate resolving resource contention issues.

#### Blocked workload implementation

The support is invoked by using two new parameters, BLWLTRPCT and BLWLINTHD, in the IEAOPTxx parmlib member to control the percentage of CPU spent for promotion.

**BLWLINTHD** This specifies the threshold time interval for which a blocked address space or enclave must wait before being considered for promotion.

If the CPU utilization of a system is at 100%, then workloads with low importance (low dispatch priority) might not get dispatched anymore. This can lead to problems if the low priority work holds a resource that is required by high priority workloads. Therefore, if an address space or enclave has ready-to-run work units (TCBs or SRBs) but does not get CPU service for the specified time interval because of its low dispatch priority, it will be temporarily promoted to a higher dispatch priority. Address spaces that are swapped out are not considered for promotion.

Minimum is 5 seconds. Maximum is 65535 seconds.

Default is 60 seconds.

**BLWLTRPCT** This specifies how much of the CPU capacity is to be used to promote blocked workloads.

This parameter does not influence the amount of CPU service that a single blocked address space or enclave is given. Instead, this parameter influences how many different address spaces or enclaves can be promoted at the same point in time. If the value specified with this parameter is not large enough, blocked workloads might need to wait longer than the time interval defined by BLWLINTHD.

Minimum is 0 units. Maximum is 200 units (=20%). (0% implies trickle should not occur.)

Default is 5 (=0.5%), max is 200 (=20%).

For detailed information about the new IEAOPTxx parameters, refer to *z/OS MVS Initialization and tuning Reference*, SA22-7592.

## 10.4 RMF enhancements for blocked workloads

RMF enhances the Workload Activity report and the CPU Activity report to provide information about the handling of blocked workload. In addition, RMF provides new overview conditions for the Postprocessor based on SMF record 72-3. This new functionality is available as SPE and needs to be installed as APAR OA18244. If the CPU utilization of a system is at 100%, workloads with low importance (low dispatch priority) might not get dispatched anymore. This could cause problems if the work holds a resource and by that holds up more important workloads. Therefore, any address space or enclave which has ready-to-run work units but does not get CPU service within a certain time interval due to its low dispatch priority will be temporarily promoted to a higher dispatch priority. RMF supports this function by reporting relevant measurements in the new Blocked Workload Promotion section of the Postprocessor CPU Activity report. The Postprocessor Workload Activity report provides the CPU time, that transactions of a certain service or report class were running at a promoted dispatching priority.

### 10.4.1 RMF CPU Activity report

The CPU Activity report is extended by a BLOCKED WORKLOAD ANALYSIS section. This new section lists the number of blocked dispatchable units that may get promoted in their dispatch priority to help blocked workloads (promote rate). This value is derived from the OPT parameters BLWLTRPCT and BLWLINTHD. To help customers tune those new OPT parameters, the new section also lists the average exploitation of the promote rate during the measurement interval. A value below 100% indicates that not all blocked workloads could be promoted. To assess the amount of workload still being blocked, the average number of address spaces and enclaves found blocked (waiters for promote) is listed. The average across the measurement interval might be quite low although there might be considerable peaks of blocked workload. Thus, the peak value detected during the measurement interval is listed as well.

#### Blocked workload analysis

The field WAITERS FOR PROMOTE, shown in Figure 10-3 on page 134, displays the “number of waiters”. This report can be used to adjust the BLWLTRPCT parameter. As long as the number of waiters is greater than 0, the system has blocked work, indicating a need to increase BLWLTRPCT. The Blocked Workload Analysis section lists the number of blocked dispatchable work units that are eligible for dispatching priority promotion. This section also lists the defined average promotion rate and the percentage used during the measurement interval. A value below 100% indicates that not all blocked workloads could be promoted. To assess the amount of workload still being blocked, the average and peak number of address spaces and enclaves found blocked is also listed and defined as follows:

<b>DEFINED</b>	Average number of blocked dispatchable work units which may get promoted in their dispatching priority per second. This value is derived from OPT parameter BLWLTRPCT.
<b>USED (%)</b>	The utilization percentage of the defined promote rate during the reporting interval.
<b>AVG</b>	Average number of address spaces and enclaves found blocked according to BLWLINTHD during the report interval.
<b>PEAK</b>	Highest number of address spaces and enclaves found blocked during the report interval.

CPU ACTIVITY									
z/OS V1R9									
CPU	2094	MODEL	714	H/W MODEL	S18				
BLOCKED WORKLOAD ANALYSIS									
OPT PARAMETERS:	BLWLRPCT (%)	0.5	PROMOTE RATE:	DEFINED	50000	WAITERS FOR PROMOTE:	AVG	0.001	
	BLWLINTHD	60		USED (%)	95		PEAK	15	

Figure 10-3 Blocked workload analysis in CPU Activity Report

If you experience a problem with blocked work holding resources for too long but you see no waiters in the RMF data, you might want to decrease BLWLINTHD. Also SMF Record 99 Subtype 1 shows the number of address spaces or enclaves waiting longer than the threshold. The field name is SMF99\_CCTRCWTR and is shown in Table 10-1.

Table 10-1 System state information section for SMF record type 99

Offsets	Name	Length	Format	Description
28 120	SMF99_CCTINTHD	2	Binary	OPT parameter BLWLINTHD starvation threshold
29 122	SMF99_CCTTRPCT	2	Binary	OPT parameter BLWLRPCT for percentage of CP trickling
31 138	SMF99_CCTRCWTR	4	Binary	Number of address spaces or enclaves waiting longer than the threshold

For more information about SMF Record 99 Subtype 1, including details of the new fields, refer to *z/OS MVS System Management Facilities, SA22-7630*.

## 10.4.2 RMF Workload Activity report

Using the RMF WLMGL report, you can analyze the following areas:

- ▶ Identify service classes running work units at a promoted dispatching priority
- ▶ Monitor the amount of CPU time that transactions of the service or report class (period) were running at a promoted dispatching priority

The amount of CPU time transactions that were running at a promoted dispatching priority is provided in the SERVICE TIMES block.

In Figure 10-4, the field PROMOTED shows the CPU time in seconds that transactions in this group were running at a promoted dispatching priority.

WORKLOAD ACTIVITY													
-TRANSACTIONS-	TRANS-TIME	HHH.MM.SS.TTT	--DASD I/O--	--- <td>--SERVICE TIMES--</td> <td>----APPL%----</td> <td>-----STORAGE-----</td> <td colspan="4"></td>	--SERVICE TIMES--	----APPL%----	-----STORAGE-----						
AVG	10.07	ACTUAL	1.00.895	SSCHRT	20.8	IOC	96199	CPU	1.400	CP	0.20	AVG	980.42
MPL	10.07	EXECUTION	59.616	RESP	2.8	CPU	266554	SRB	0.200	AAPCP	0.00	TOTAL	9868.59
ENDED	1	QUEUED	1.278	CONN	0.2	MSO	9131	RCT	0.000	IIPCP	0.00	SHARED	127.67
END/S	0.00	R/S AFFIN	0	DISC	0.0	SRB	39610	IIT	0.200				
#SWAPS	30	INELIGIBLE	0	Q+PEND	0.5	TOT	411494	HST	0.000	AAP	0.00	--PAGE-IN RATES--	
EXCTD	0	CONVERSION	0	IOSQ	2.0	/SEC	457	AAP	0.000	IIP	N/A	SINGLE	0.0
AVG ENC	0.00	STD DEV	0					IIP	N/A			BLOCK	0.0
REM ENC	0.00					ABSRPTN	45					SHARED	0.0
MS ENC	0.00					TRX SERV	45	<b>PROMOTED</b>	<b>0.333</b>			HSP	0.0

Figure 10-4 Field PROMOTED in Workload Activity Report

### 10.4.3 New SMF record types

SMF record type 70 subtype 1 and 72 subtype 3 are handled consistently within RMF. New fields are appended to the end of the respective data sections. If SMF records without the new fields are formatted by the RMF Postprocessor, the appropriate report fields are reported with ZERO values. No compatibility PTFs are required. This support is rolled back as SPE APAR OA18244.

The CPU control section is updated with information about blocked workloads as described in Table 10-2. The RMF Postprocessor CPU activity report uses this information to format the BLOCKED WORKLOAD ANALYSIS section.

Table 10-2 SMF record type 70 subtype 1 (CPU Activity) – CPU control section

Offset	Name	Length	Format	Description
5 5	SMF70STF	1	Binary	Flag BIT 0 to 4 meaning not changed. BIT 5 = OPT parameter BLWLTRPCT changed. BIT 6 = OPT parameter BLWLINTHD changed BIT 7 = reserved
96 60	SMF70PMI	4	Binary	Accumulated number of blocked dispatchable units per second that may get promoted in their dispatch priority. To get the average promote event rate, divide SMF70PMI by SMF70SAM.
100 64	SMF70PMU	4	Binary	Number of blocked dispatchable units being promoted during the interval
100 64	SMF70PMW	4	Binary	Accumulated number of address spaces and enclaves being blocked during the interval. To get the average number of waiters for promote, divide SMF70PMW by SMF70SAM.
108 6C	SMF70PMP	4	Binary	Maximum number of address spaces and enclaves found being blocked during the interval.
112 70	SMF70PMT	2	Binary	1/1000s of a CP for promote slices (OPT parameter BLWLTRPCT).
114 72	SMF70PML	2	Binary	Swapped-in starvation threshold. When an address space or enclave has not received CPU service within this time interval, it is considered being blocked (OPT parameter BLWLINTHD).

The service or report class period data section of SMF record type 72-3 is extended with a new field holding the CPU time at promoted dispatching priority to help blocked workloads, and it is shown in Table 10-3. This field is used to format the PROMOTED field in the Resource Consumption Section of the WLM Workload Activity Report.

Table 10-3 SMF record type 72 subtype 3 (Workload Activity) – SC/RC period data section

Offset	Name	Length	Format	Description
576 240	R723TPDP	8	Float	Total CPU time at promoted dispatching priority (in 1024 microsecond units)

In addition, two new overview control statements are provided based on this new SMF record type 72-3 fields, as shown in Table 10-4 on page 136.

Table 10-4 New overview conditions based on SMF record 72-3

Condition	Condition Name	Source	Algorithm
Percentage of CPU time at promoted dispatching priority	PROMPER	R723TPDP	Sum(R723TPDP) / Interval length x 100
Time at promoted dispatching priority in seconds	PROMSEC	R723TPDP	Sum(R723TPDP)

**Note:** Both overview conditions can be specified with a service class (period), report class (period), workload or the POLICY qualifier.

#### 10.4.4 RMF Distributed Data Server

There are new metrics for the RMF Distributed Data Server. For resource type MVS\_IMAGE, the CPU time at promoted dispatching priority is provided by:

- ▶ WLM workload
- ▶ WLM service class and service class period
- ▶ WLM report class and report class period

Exploit the new metrics with RMF PM or the RMF Monitor III Data Portal for z/OS.

### 10.5 Improved assist processor routing services

The zSeries platform recently introduced specialty processors that can be deployed and exploited by qualified workloads on z/OS. This includes support for the following processors:

- ▶ zAAP (zSeries Application Assist Processor)

These processors can be used for JAVA application workloads on z/OS (including workloads running under z/OS WebSphere Application Server).

- ▶ zIIP (System z Integrated Information Processor)

These processors can be used by qualified z/OS DB2-related workloads.

When the Sysplex Distributor routes incoming connections based on Workload Manager (WLM) system weights (in addition to using WLM information about general CPU capacity), it will optionally (if all systems in the sysplex are V1R9 or later) consider available zAAP CPU capacity, available zIIP CPU capacity, or both. When configured on the VIPADISTRIBUTE statement, a composite WLM weight is determined based on the available capacity of each processor type and the expected use of each processor type by this application.

The sysplex routing services provide routing recommendations to help distributed programs make the routing decisions. The recommendations are based on processor capacities. The sysplex routing services allow work associated with a server to be distributed across a sysplex. They enable distributed client/server environments to balance work among multiple servers.

With zAAPs and zIIPs two new processor types are introduced for z/OS systems. It is now possible that a substantial amount of work is executed on the assist processors in addition to



regular CPs. In previous releases, the routing recommendation is based on the capacity of regular CPs only.

With z/OS Version 1 Release 9, routing services provide individual weights for all processor types and also a combined server weight based on usage of all processors. The benefit is better routing recommendations in environments with assist processors.

Prior to z/OS V1R9, the routing services only return weights based on capacity of regular CPs, but the capacity of assist processors influence how the work can be processed. Routing services now provides individual weights for all processor types and also a combined server weight based on usage of all processors. This provides better routing recommendations in environments with assist processors such as zIIPs and zAAPs.

### 10.5.1 Sysplex routing services IWMSRSRS improvements

In previous releases the routing recommendation is based on the capacity of regular CPs only. With z/OS V1R9, routing recommendations now take the capacity of assist processors (zAAPs and zIIPs) into account.

This new support is invoked by the following WLM services:

- ▶ IWMSRSRS FUNCTION=SELECT or IWMSRSRS FUNCTION=SPECIFIC  
The existing calls now return a list of registered servers in a sysplex along with 4 weight values for each server (3 individual, 1 combined) which tell the caller the relative number of requests to send to the server
- ▶ IWM4SRSC  
This existing call now returns the weight(s) for one specific server that is identified by its STOKEN.
- ▶ IWMWSYSQ EXTENDED\_DATA=YES  
This existing call returns capacity information for all processor types of all systems in the sysplex.

For more information about WLM services, refer to *MVS Programming: Workload Management Services*, SA22-7619.

#### IWMSRSRS service

The IWMSRSRS service, shown in Figure 10-5 on page 138, provides three functions: SELECT, QUERY and SPECIFIC. This call returns a list of registered servers known to the system on which the service is invoked.

When either the SELECT or the SPECIFIC function is chosen, IWMSRSRS returns a list of servers in the sysplex which are associated with the input location name along with a relative weighting for each server. These servers are identified by their Network ID and LU name, which were previously registered using the sysplex router register service, IWMSRSRG, which allows a caller to register as a server.

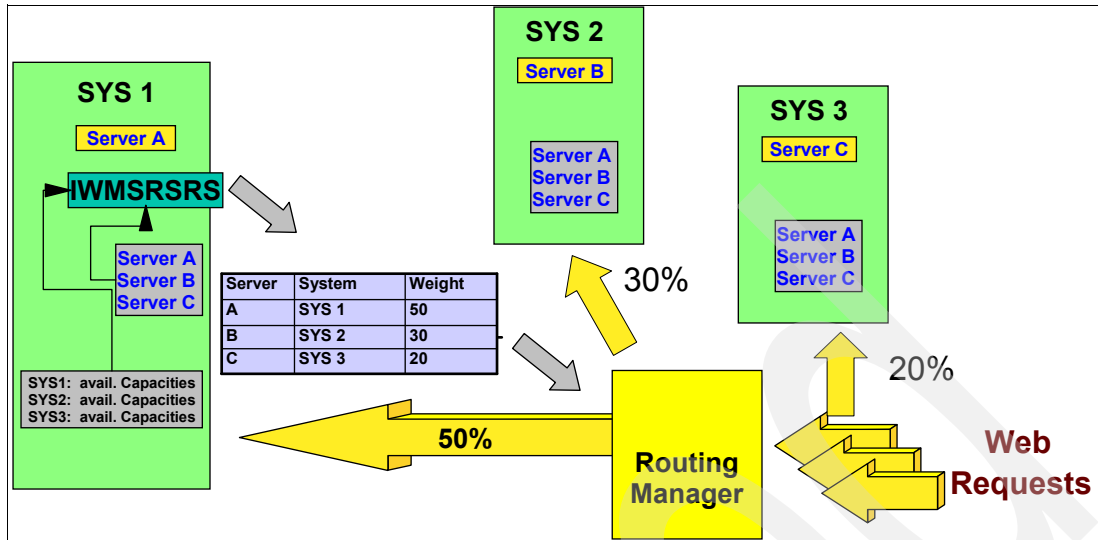


Figure 10-5 IWMSRSRS service

### IWMSRSRS routing recommendation

Each system in the sysplex has information about all registered servers and all processor capacities. To get a routing recommendation, you can call the IWMSRSRS service with a LOCATION parameter on any system in the sysplex. It returns recommendations, called *weights* (numbers between 0 and 64) for each server that was registered under that LOCATION ID. Then you would use those weights to distribute the incoming request between the servers according to the size of their weights, as shown in Figure 10-5. The base calculation for FUNCTION=SPECIFIC and FUNCTION=SELECT is the same as in previous releases.

In previous releases:

- ▶ Only one weight (SYSR\_WEIGHT) is returned and it is based only on regular CP capacity.
- ▶ A system was *not* returned if it had less than 5% of displaceable capacity at the selected importance level

With z/OS Version 1 Release 9:

- ▶ SYSR\_WEIGHT now is the combined weight of all available processor resources.
- ▶ The individual weights have been added:
  - SYSR\_CPU\_WEIGHT (corresponds to SYSR\_WEIGHT of previous releases)
  - SYSR\_ZAAP\_WEIGHT
  - SYSR\_ZIIP\_WEIGHT
- ▶ A system is *not* returned if it has no displaceable capacity for regular CPs at the selected importance level
  - As a result, potentially more systems may be returned.
  - If a system has no displaceable capacity for an assist processor, it is returned and missing capacity is reflected in the weight.

### IWMSRSRS details - Step 1

The IWMSRSRS service calculates the CP system weight to find the importance level where at least 5% of displaceable capacity is available. This results in an importance level 5. A

shown in Figure 10-6, systems B and C meet the criteria. As shown in Figure 10-7, however, system A does not meet the criteria.

<b>Selected Importance Level</b>	=	<b>5</b>
<b>Total capacity at level 5</b>	<b>40+390+700 =</b>	<b>1130</b>
<b>CPU System Weight for System A</b>	<b>40*64/1130 =</b>	<b>2</b>
<b>CPU System Weight for System B</b>	<b>390*64/1130 =</b>	<b>22</b>
<b>CPU System Weight for System C</b>	<b>700*64/1130 =</b>	<b>40</b>

Figure 10-6 Results from the calculation of the system weights

**Note:** Prior to z/OS V1R9, system A would have been removed from the list of eligible systems. Now with the new changes, system A is still considered because the consumption is *not* zero.

Level	System A		System B		System C	
	SUs	%	SUs	%	SUs	%
<b>0 (System)</b>	2000	100	3000	100	2000	100
<b>1</b>	1800	90	2700	90	1700	85
<b>2</b>	1400	70	2100	70	1500	75
<b>3</b>	600	30	1800	60	1400	70
<b>4</b>	160	8	900	30	900	45
<b>5</b>	40	2	390	13	700	35
<b>6 (Disc)</b>	0		120	4	60	3
<b>7 (Free)</b>	0		30	1	0	

Figure 10-7 Capacity for regular CPs

### IWMRSRS details - Step 2, 3, and 4

With z/OS V1R9, three new output weights have been introduced: the CPU weight, the zAAP weight and the zIIP weight. The CPU weight is computed the same way as the weight prior to V1R9, taking only CPU data into account. The zAAP and the zIIP weights are computed taking only zAAP, respectively zIIP, data into account. The weight (also referred to as “mixed” weight) is a combination of these three processor weights.

With z/OS V1R9, the new support from this service now attempts to include special purpose processors in the calculation of the weights, as explained here.

- ▶ Step 2
  - Calculate the system weights for the assist processors.
  - Systems which have been excluded in step 1 are no longer considered.
  - In this step, no system is excluded.
- ▶ Step 3
  - Calculate the server weight.
  - Use the proportion of the work using the processor types to calculate a system weight for the server and scale the resulting weight to 64.

- ▶ Step 4
  - Include other weight factors (zAAPs and zIIPs).

The results are shown in Figure 10-8.

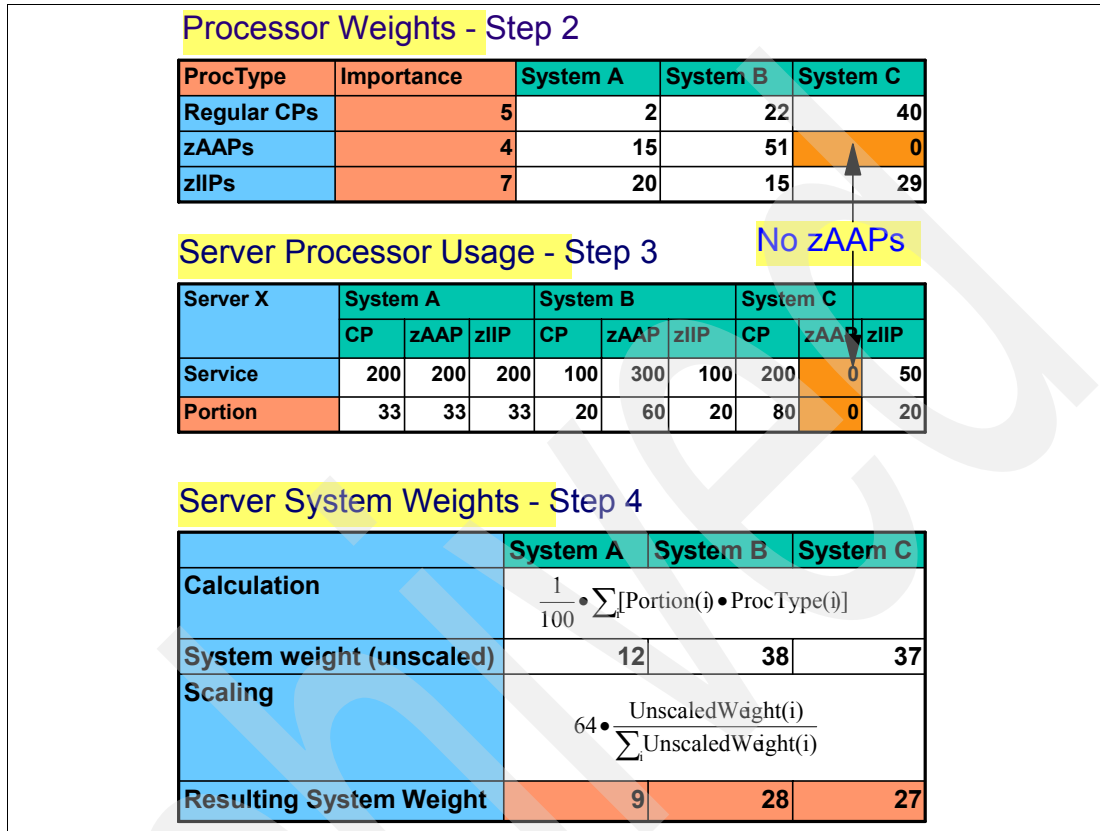


Figure 10-8 Calculation of the weights

### Server weights on System C

During Step 4, the following calculations for system C servers are as explained here.

- ▶ Calculate the server weights.
- ▶ Divide the system weights by the number of servers on system.
- ▶ Only for function=SPECIFIC: include other server performance factors.
- ▶ In case one of the servers has a mixed weight=0: return only 1 server (and undivided weight): the server with maximum performance.

The results are shown in Figure 10-9 on page 141.

Server Weights - Step 4									System C			
Server	Server C_1				Server C_2				Server C_3			
Processor	W	CPU	zAAP	zIIP	W	CPU	zAAP	zIIP	W	CPU	zAAP	zIIP
		W	W	W		W	W	W		W	W	W
System Weight	27	40	0	29	27	40	0	29	27	40	0	29
# of Servers	3											
Divided weights	9	13	0	10	9	13	0	10	9	13	0	10
Server performance factors (PI, Queue)	100%				25%				66%			
Server Weights	9	13	0	10	4	6	0	5	6	8	0	6

Figure 10-9 Server weight calculation for System C

### IWMSRSRS additional enhancements

The restriction that no more than 96 servers per system could be returned is now relieved. With z/OS V1R9, up to 300 servers per system will be returned from the IWMSRSRS service. This is also valid for IWMSRSRS FUNCTION=QUERY service calls.

This support is available for z/OS V1R6 and above via APAR OA18531. In addition, the C interface to IWMSRSRS IWMDNSRV has also been extended to return the new weights.

## 10.5.2 Sysplex routing services IWM4SRSC improvements

The IWM4SRSC service provides information about how well a server is suitable to receive work from a WLM point of view. The IWM4SRSC service allows you to check a specific server before routing work to it from WLM. Thus, the information obtained can be used for making balanced routing decisions with programs like Sysplex Distributor and their exploiters (for example, TCP/IP).

The input to the IWM4SRSC service is the STOKEN of an address space. The output is an indicator of how well the address space itself and the transactions or enclaves (if it is a registered transaction server, an enclave server or an enclave owner) are performing relative to their WLM goal and to the displaceable capacity for its WLM importance on that system.

The service returns an indicator that can be used for load balancing by comparing it to calls of this service for other servers. The indicator output is a weight. This weight is calculated based on six factors, as follows:

- ▶ It is a combination of the three processor weights (CPU weight, zAAP weight and zIIP weight) and
- ▶ The respective consumed service units (CPU weight, zAAP weight and zIIP weight)

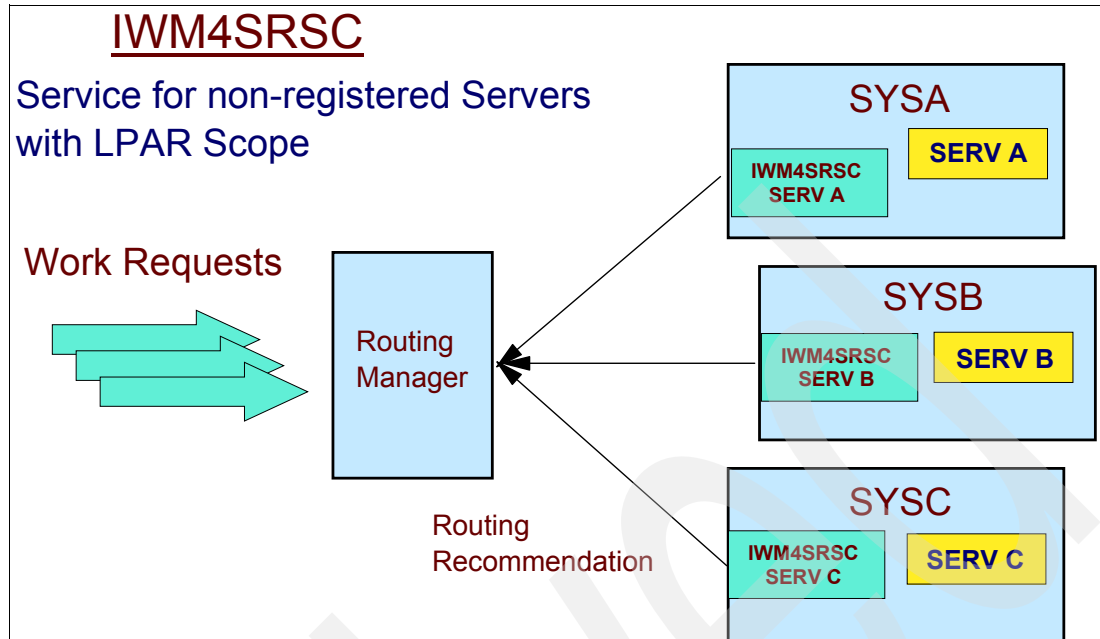


Figure 10-10 IWM4SRSC service for non-registered servers with LPAR scope

### Weight calculation

The CPU, zAAP and zIIP weights are each computed based on the following four factors:

- ▶ The first factor is how well this server, or the transactions or enclaves it is related to, fulfill their goals.
- ▶ The second factor is how much other work with lower importance can be displaced, if it receives more work to handle on this system.
- ▶ The third factor is the abnormal termination factor. This depends on the ratio of abnormal terminations to normal terminations as reported by the IWMRPT service. If no terminations were reported by IWMRPT, this factor is neutral (=1).
- ▶ The fourth factor is the health factor of this server. It is dependent on the health indicator which was reported to WLM for this server by the IWM4HLTH service or by IWMSRSRG. If no health indicator was reported, this factor is also neutral.

### Create output weight as a number

These four factors are combined to create the output weight as a number.

To make it easier for the caller to determine, how far the weights were influenced by the abnormal terminations and health factors, those values can also be output through the optional parameters ABNORM\_COUNT and HEALTH.

The processor weights are returned through the optional CPUWEIGHT, ZAAPWEIGHT, and ZIIPWEIGHT parameters. The respective parts of these weights in the WEIGHT are returned through the optional parameters CPUPROPORTION, ZAAPPROPORTION, and ZIIPPROPORTION.

The WEIGHT is equal to the sum of these three proportion fields. As WLM computes the values with higher precision, and rounds them before output, the WEIGHT actually returned is probably greater than the sum of the returned proportion fields by one or two units.

## IWM4SRSC new and changed parameters

There are required output parameters, which contain the weight of how well the server is performing. When all systems in the sysplex are running z/OS V1R9, the weight is based on capacities of all processor types. Otherwise, WEIGHT and CPUWEIGHT will be identical. The parameters are as follows:

### **,CPUWEIGHT=cpuweight**

An optional output parameter, which contains the weight of how well the server is performing on regular CPs. This is equivalent to the old weight returned on z/OS V1R8 or before.

### **,ZAAPWEIGHT=zaapweight**

An optional output parameter, which contains the weight of how well the server is performing on ZAAPs. ZAAPWEIGHT will return 0 if there is at least one systems in the sysplex with a z/OS level prior to V1R9.

### **,ZIIPWEIGHT=ziipweight**

An optional output parameter, which contains the weight of how well the server is performing on ZIIPs. ZIIPWEIGHT will return 0 if there is at least one systems in the sysplex with a z/OS level prior to V1R9.

### **,CUPROPORTION=cuproportion**

An optional output parameter. When all systems in the sysplex are running z/OS V1R9 the returned WEIGHT value (specified above) is a composite value that is based on the amount of each processor type that a server is consuming. The returned value of this parameter will be the proportion of CPUWEIGHT that was used to determine the composite WEIGHT for this application.

### **,ZAAPPROPORTION=zaaproportion**

An optional output parameter. When all systems in the sysplex are running z/OS V1R9 the returned WEIGHT value (specified above) is a composite value that is based on the amount of each processor type that a server is consuming. The returned value of this parameter will be the proportion of ZAAPWEIGHT that was used to determine the composite WEIGHT for this application.

### **,ZIIPPROPORTION=ziiproportion**

An optional output parameter. When all systems in the sysplex are running z/OS V1R9 the returned WEIGHT value (specified above) is a composite value that is based on the amount of each processor type that a server is consuming. The returned value of this parameter will be the proportion of ZIIPWEIGHT that was used to determine the composite WEIGHT for this application.

The returned weights are a number between 0 and 64.

## 10.5.3 IWMWSYSQ service

The purpose of this service is to query information about the systems in the sysplex that are in goal mode. The query system information service, IWMWSYSQ, returns a list of systems running in goal mode and information related to available CPU capacity and resource constraints.

The enhancement for z/OS V1R9 is available via a new parameter EXTENDED\_DATA where:

- ▶ EXTENDED\_DATA=YES

Using this option allows additional information to be returned in the output area, as follows:

- The system level which contains the total system capacity has been added.
  - Data is returned for all processor types.
  - In addition the following information is supplied:
    - Uniprocessor speed of a single processor.
    - zAAP and zIIP normalization factors (deviation from regular processor speed if applicable).
- ▶ EXTENDED\_DATA=NO

Using this option returns the output area the same as in previous releases

## 10.5.4 Migration and coexistence considerations

When pre-z/OS V1R9 systems are active in a sysplex, the weights are based on regular CP capacity as before because there is no information available about zAAP and zIIP capacities from the older releases.

In a z/OS V1R9 system, you automatically get the new combined server weights without changes to the service invocation.

## 10.6 Group capacity limit

In z/OS V1R8 and z/OS V1R9, the WLM defined capacity mechanism is extended to handle LPAR groups instead of a single LPAR. This is called *group capacity limit* support. Since the use of defined capacity began some years ago, there is a requirement to have more flexibility when defining capacity limits for LPARs, so an enhanced mechanism is needed.

The group capacity limit balances the capacity between groups of partitions on the same processor. This requirement, related to WLM, requires the use of IBM System z9 and z/OS V1R8 and higher. The software support allows grouping of LPARs in the same processor and the LPARs are then managed on a group basis using the existing WLM defined capacity mechanism.

### 10.6.1 Defined capacity review

The enhancement to WLM to support group capacity limits requires an understanding of all the concepts of, processors in LPAR mode, shared logical CPs, LPAR dispatching, LPAR weights, and the 4-hour rolling average.

#### Defined capacity

As part of the z/OS support of Workload License Charges, you can set a defined capacity limit, also called a *soft cap*, for the work running in a logical partition. This defined capacity limit is measured in millions of service units per hour (MSUs). It allows for short-term spikes in the CPU usage, while managing to an overall, long-term, rolling 4-hour rolling average. It applies to all work running in the partition, regardless of the number of individual workloads the partition may contain.

#### LPAR weights

LPAR weights are used to control the distribution of shared CPs between LPs. Therefore, LPs with dedicated CPs do not use LPAR weights.



LPAR weights determine the guaranteed (minimum) amount of physical CP resource an LP should receive (if needed). This guaranteed figure may also become a maximum when either:

- ▶ All the LPs are using all of their guaranteed amount (for example, if all LPs were completely CPU-bound).
- ▶ The LP is capped using traditional LPAR capping.

An LP may use less than the guarantee if it does not have much work to do. Similarly, it can use more than its weight if the other LPs are not using their guaranteed amount.

## LPAR LIC

LPAR LIC uses *weights* and the *number of logical CPs* to decide the priority of logical CPs in the logical CP ready queue. The following formulas are used by LPAR LIC in the process on controlling the dispatching of logical CPs:

- ▶  $WEIGHT(LP_x)\% = 100 * WEIGHT LP_x / SUM\_of\_ACTIVE LPs WEIGHTs$

This indicates the percentage of the total shared physical CP capacity that will be guaranteed to this LP. This percentage will vary, depending on the weights of all the active LPs.

- ▶  $TARGET(LP_x) = WEIGHT(LP_x)\% * (\# \text{ of } NON\_DEDICATE\_PHYS\_CPs)$

This indicates, in units of shared physical CPs, how much CP resource is guaranteed to the LP. This figure cannot be greater than the number of logical CPs in the LP. This is because you cannot use more physical CPs than the number of logical CPs you have defined—each logical CP can be dispatched on only one physical CP at a time. So, even if there are eight physical CPs available, an LP that has been defined with only four logical CPs can only ever use four of the physical CPs at one time. If you specify a weight that guarantees you more capacity than can be delivered by the specified number of logical CPs, the additional unusable weight will be distributed among the other LPs.

- ▶  $TARGET(LCP_x)\% = TARGET(LP_x) / (\# \text{ of } LCPs\_in\_LP_x) * 100$

This takes the  $TARGET(LP_x)$  value (that is, the number of physical CPs of capacity) and divides that by the number of logical CPs defined for the LP. The result determines the percentage of a physical CP that should be given to each logical CP. This in turn determines the effective speed of each logical CP.

Over time, the average utilization of each logical CP is compared to this value. If Target is less than Current, then the logical CP is taking more CP resource than the guarantee and its priority in the ready queue is decreased. It does *not* mean that it is prohibited from consuming CP; it simply means that it will tend to sit lower in the queue than other logical CPs that have used less than their guaranteed share of the CP resource.

Also, these logical CPs are going to be preemptable by an I/O interrupt for a logical CP that is behind its target. If Target is greater than Current, then the logical CP is taking less CP resource than the guarantee and its priority in the ready queue is increased. This means that it has a better chance of being dispatched on a physical CP. Also, these logical CPs are not going to be preempted by an I/O interrupt for another logical CP.

## 4-hour rolling average

WLM enforces the defined capacity limit by tracking the partition's CPU usage and continually averaging it over the past 4 hours. Spikes above the defined capacity limit are possible, as shown in Figure 10-11 on page 146 with the dashed line, as long as they are offset by low points that keep the 4-hour average at or below the defined capacity limit. When this 4-hour average goes over the defined capacity limit, then WLM caps the partition (soft cap). At that point, it can use no more than the defined capacity limit, until the average drops below the limit.

At IPL, WLM defaults to a 4-hour time interval that contains no partition CPU usage. This allows the defined capacity limit to be exceeded at IPL and therefore avoids capping during this time.

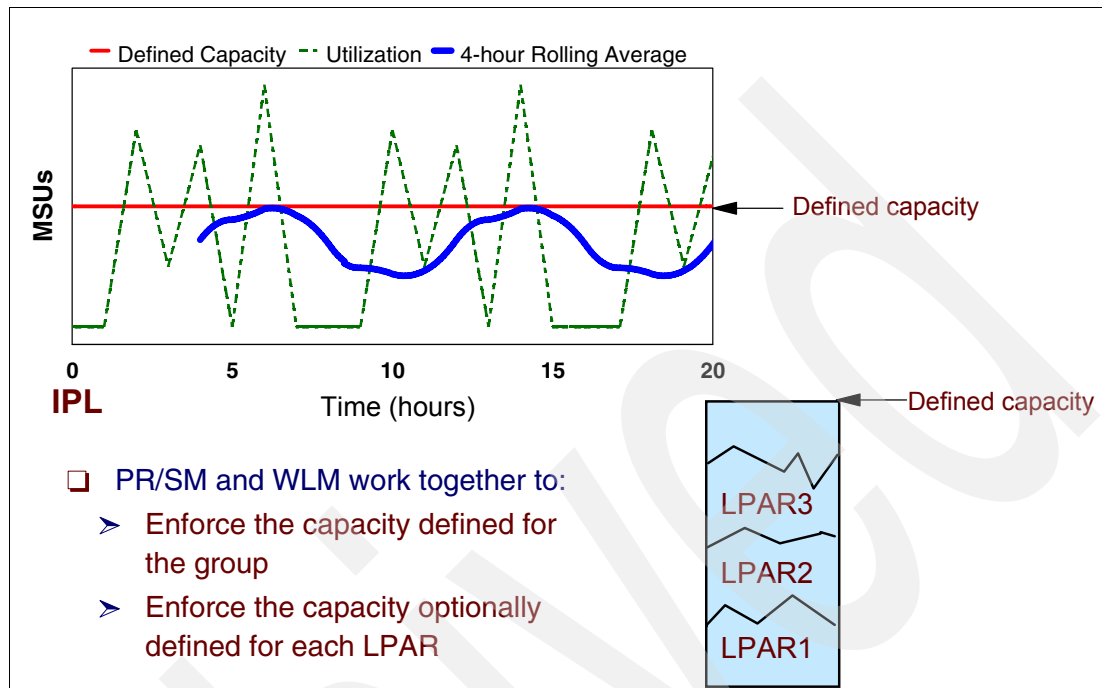


Figure 10-11 Defined capacity example showing 4-hour rolling average with three LPARs

## 10.6.2 Group capacity definition rules

Each partition manages itself independently from all other partitions and the group capacity you define is based on defined capacity. Therefore, a 4-hour rolling average of the group MSU consumption is used as a base for managing the partitions of the group.

Each partition is going to see the consumption of all the other LPARs on the processor. If the partition belongs to a group, it identifies the other partitions in the same group and calculates its defined share of the capacity group based on the partition weight (compared to the group). This share is the target for the partition if all partitions of the group want to use as much CPU resources as possible.

If one or more LPARs do not use their share, this donated capacity is distributed over the LPARs which need additional capacity. Even when a partition receives capacity from another partition, it never violates its defined capacity limit (if one exists).

### Defining group capacity limits

A capacity group has to be defined using the following rules:

- ▶ Consists of multiple LPARs on the same processor.
- ▶ LPARs must run at least z/OS V1R8 or higher.
- ▶ It is possible to define multiple groups on a processor.
- ▶ A partition can only belong to one group.
- ▶ A group member can have a defined capacity also.
- ▶ A capacity group is independent of a sysplex and an LPAR cluster.

### 10.6.3 Group capacity example

This implementation should make the use of use soft capping easier to utilize CPU capacity across multiple logical partitions on the same processor. With group capacity limits, it is no longer necessary to worry about the SCRT reporting period in order to balance the defined capacity across two LPARs. In the following example:

- ▶ Assume a processor with a total capacity of 500 MSU and two partitions named PROD and TEST in which the MSU consumption should be controlled via soft capping.
- ▶ LPAR PROD should use a defined capacity limit of up to 300 MSU and LPAR TEST of up to 100 MSU.
- ▶ During certain periods LPAR PROD needs more capacity while at the same time LPAR TEST does not need all of its capacity.

#### Current LPAR support

Without group capacity, you must now perform the following steps to ensure that the SCRT reporting agrees with the IBM contract in order to avoid additional charges:

1. Reduce the defined capacity limit of partition TEST by 50 MSU. The new defined capacity limit is now 50 MSU for TEST.
2. Wait for one hour to assure that the SCRT reporting period contains this change.
3. Now you can increase the defined capacity limit for partition PROD to 350 MSU.

This requires some planning in advance.

#### Using group capacity support

With group capacity, the situation becomes much easier because you must define a group for the LPARs PROD and TEST and a group limit of 400 MSU. Now you have two options regarding how you can configure your partitions to ensure that a CPU consumption of 300 MSU is guaranteed for PROD and 100 MSU for TEST, as follows:

1. Ensure that the partition weights for PROD and TEST entitle partition PROD to a  $\frac{3}{4}$  share and TEST to a  $\frac{1}{4}$  share. For example, set the weights to 300 for PROD and 100 for TEST. So the actual weight setting must confirm your objectives with other partitions which might run on the system. With such a setting, PROD is now able to obtain  $\frac{3}{4}$  of the 400 MSU and TEST  $\frac{1}{4}$  of it. During periods where one partition uses much less capacity than the other partition, that LPAR is able to get up to 400 MSU.
2. If you want to ensure that PROD can get everything and TEST not more than the 100 MSUs, you can also set an individual defined capacity limit of 100 MSU for TEST. Now TEST is only able to receive 100 MSU while PROD gets always at least 300 MSU and can use up to 400 MSU if TEST does not need the capacity.

Both examples assume that the partition weights of all partitions on the processor allow PROD and TEST to consume the desired amount of service.

### 10.6.4 Hardware and software for group capacity

Group capacity limit is available with z/OS V1R8. The hardware requires is an IBM System z9 with a new microcode level.

Using the HMC or SE of the new microcode level installed processor, there are two new input values in the LPAR configuration panel. These are “Group Name” and “Group Limit”, as shown in Figure 10-12 on page 148.

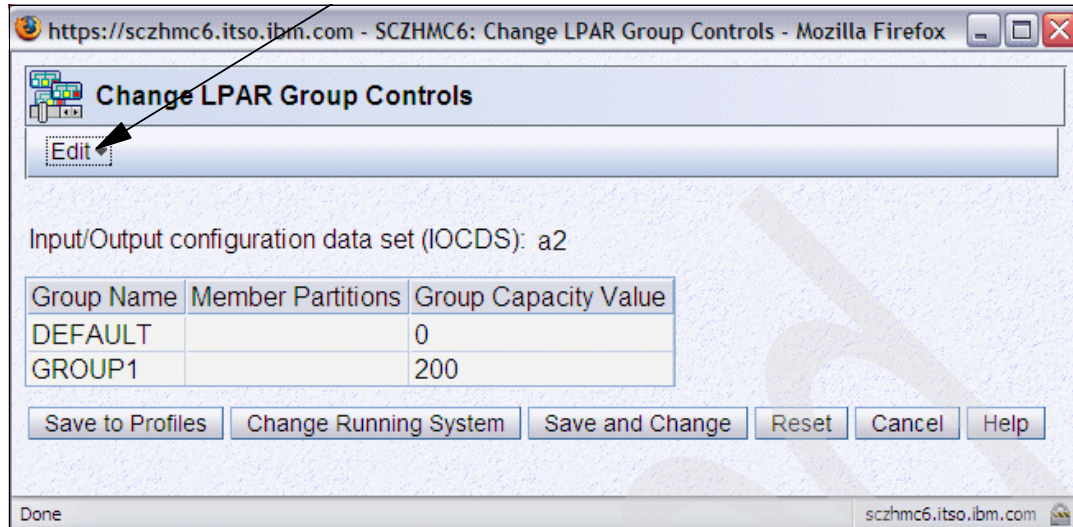


Figure 10-12 HMC LPAR configuration panel allowing group capacity limit definitions

## New HMC panels

The group name and limit field are available in processor activation Profile and also in Change Dynamic LPAR configuration panel.

Customers can define a capacity group by entering the group name and group limit value for the partitions which should belong to the same group. The group limit definition is independent from the defined capacity definition. Both limits can be defined and both work together.

## Group capacity considerations

Here are the other hints and tips of new group capacity limit mechanism

- ▶ Capacity group consists of multiple LPARs on the same processor.
- ▶ All LPARs that exist in a group must run z/OS V1R8; otherwise, the group limit may not be enforced correctly.
- ▶ WLM only manages partitions with shared CPs. Dedicated partitions and partitions with weight completion equal to YES are ignored. If they have been defined to a group, they will be excluded from the group. Only the partitions with wait completion equals NO and shared processors are managed towards the capacity limit.
- ▶ It is possible to define multiple groups on a processor
- ▶ A capacity group is independent of a sysplex and an LPAR cluster. Your test system and prod system can be in the same group if there are running in the same processor.
- ▶ A partition can only belong to one group
- ▶ Each z/OS system manages itself independently from all the other partitions.
- ▶ Group capacity is based on defined capacity. Therefore, a 4-hour rolling average of the group MSU consumption is used as base for managing the partitions of the group.
- ▶ Each partition sees the consumption of the other partitions on the processor. It identifies the other partitions that exist with it on the same group. Each partition calculates its defined share of the capacity group based on the partition weight. This share is target for the partition if all partitions of the group want to use as much CPU resources as possible. If one or more LPARs do not use their share, this donated capacity will be distributed over the LPARs which need additional capacity that are in the same group. Even when a

partition receives capacity from another partition, it never violates its defined capacity limit if one exists.

## 10.6.5 Group capacity limit example

In the example shown in Figure 10-13, there are five z/OS partitions in the processor. Partitions A, B, and C belong to group1. Partitions D and E do not belong to any group. The limit that is defined for group1 is 200 MSUs. The processor has a higher capacity than 200 MSUs.

Partition	Group Name	Group Limit [MSU]	Weight	Target MSU consumption based on Weight	Defined Capacity (Softcap) [MSU]
A	Group1	200	70	93	n/a
B	Group1		50	67	80
C	Group1		30	40	30
D	n/a	n/a	100	n/a	120
E	n/a	n/a	50	n/a	n/a

Figure 10-13 Sample processor LPAR configuration

The weights that have been defined for each partition make the target MSU consumption based on the weight the values shown in Figure 10-13.

### Partition A

Partition A can use up to 200 MSUs based on the 4-hour rolling average. If all three partitions want to use as much as possible, partition A will only get up to 93 MSUs.

### Partition B

Partition B can use up to 80 MSUs because an individual soft cap is defined. If all three partitions want to use as much as possible, partition B can only get up to 67 MSUs.

### Partition C

Partition C can use up to 30 MSUs because the defined capacity is smaller than the target based on partition weight. If the other partitions does not need more, partition C cannot use more than 30 because of its individual soft cap definition of 30 MSUs.

### Group capacity after an IPL

Considering the defined capacity 4-hour rolling average startup after an IPL, the group-wide capping starts when the 4-hour rolling group average reaches the group limit.

When a system is IPLed and joins a group, it does not have the history of the MSU consumption of the complete group. Therefore, it can take up to 4 hours until all systems in the group have the same view. During that time period, the group limit cannot be guaranteed. The group might use more than the group limit because the new partition does not have the complete history. If this happens, the other partitions will be reduced in their capacity based on their weight in the group.

### Removing a member from the group

A partition can be dynamically removed from a group. A partition can be dynamically added to another group. In this situation, the changed partition has no knowledge about unused capacity of the new group, and also it does not keep a history from its previous activity. Therefore, all systems must again learn the new situation and the limit cannot be guaranteed.

## Group capacity limit and IRD

Group capacity limits can work together with Intelligent Resource Director (IRD) weight management. IRD weight management may change the weight of partitions in a capacity group and thus change the target share of the partition in the capacity group.

Group capacity limits can also work together with the IRD **VARY CPU ON/OFF** command management. Both functions should have no influence on each other.

**Note:** The Intelligent Resource Director extends the concept of goal-oriented resource management by allowing you to group system images that are resident on the same physical server running in LPAR mode, and in the same parallel sysplex, into an “LPAR cluster.” This gives Workload Management the ability to manage processor and channel subsystem resources, not just in one single image but across the entire cluster of system images.

### 10.6.6 RMF and SMF updates to support group capacity limit

The RMF Monitor I CPU Activity Report and Monitor III CPC Capacity Report are changed to support group capacity limits. New Overview conditions are also added related to the usage of group capacity limit. The Group Capacity Report example shown in See Figure 10-14 gives all the information needed to understand how the MSUs are used in the group. For more detailed information about RMF enhancements related to group capacity limits, refer to Figure 10-14.

GROUP CAPACITY REPORT										
z/OS V1R8			SYSTEM ID	RMF1	DATE 04/12/2005					
			RPT VERSION	V1R8	RMF	TIME 16.00.00				
GROUP-CAPACITY NAME	LIMIT	PARTITION	SYSTEM	-- MSU --		WGT	-CAPPING-		- ENTITLEMENT -	
				DEF	ACT		DEF	WLM%	MINIMUM	MAXIMUM
CGRP010	200	RMF1	RMFSYS01	0	122	90	NO	3.0	128	200
		RMF2	RMFSYS01	70	70	50	NO	0.0	70	70
-----				TOTAL		192	140	-----		
CGRP0200	250	DOM1	DOMSYS01	40	38	20	NO	2.2	23	40
		DOM2	DOMSYS02	0	130	100	NO	0.0	117	250
		DOM3	DOMSYS02	30	30	25	NO	0.0	29	30
		DOMIN010	DOMSYS03	0	50	25	NO	0.0	29	250
-----				TOTAL		248	170	-----		

Figure 10-14 RMF Monitor I Group Capacity Report

### SMF records

The SMF type 70 record contains group name, group membership and group MSU limit. The SMF type 99 record trace data has a new subtype 11 to analyze group capacity data.

### 10.6.7 Examples related to usage of group capacity limit

Following are two different examples related to use of group capacity limit. In the first example there are partitions with individual defined capacities. In the second example, there are partitions which have no individual defined capacity.

### Example 1

In the first example, usage of group capacity limits are with partitions that have individual defined capacities. There are three LPARs (IRD3, IRD4 and IRD5) in the same processor. They are grouped and have a group capacity limit of 50 MSUs. The individual defined capacities and weights are shown in Figure 10-15. Following the period after the IPL, you can see the usage or bonus given to IRD4 and IRD5. Following this initial period of the IPF, the partitions are then capped to 5 MSUs.

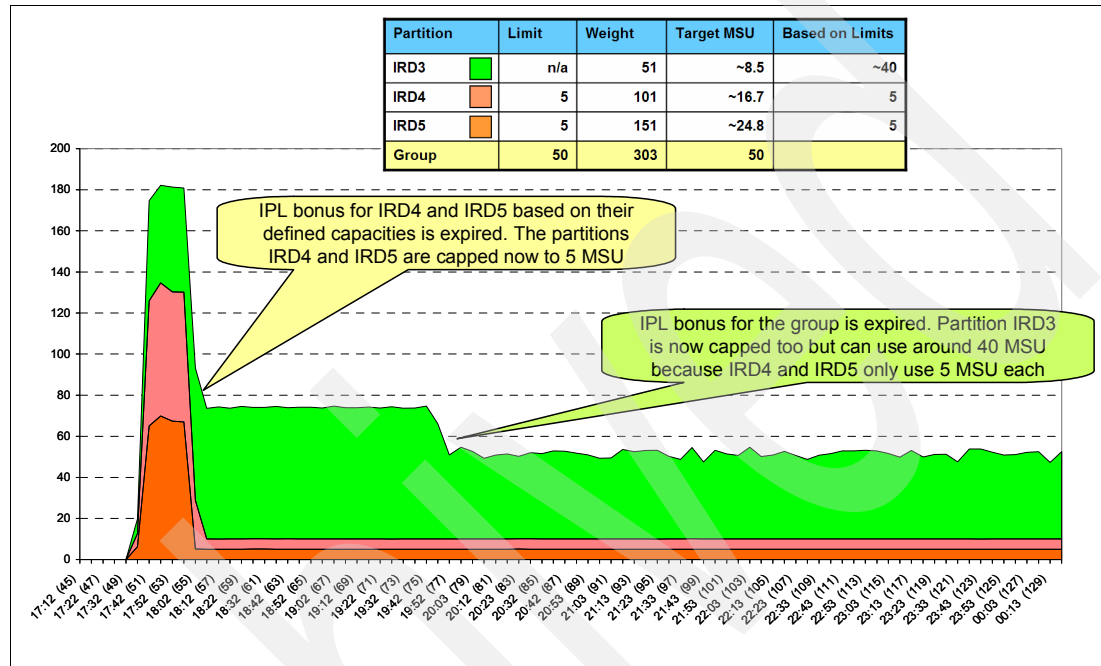


Figure 10-15 Group capacity limit usage for example 1

### Example 2

The second example shows the group capacity limit effects on partitions without individual defined capacities. IRD3, IRD4, and IRD5 are the LPARs. Their weights and group limit are shown in Figure 10-16 on page 152, along with the actual MSU values consumed by the LPARs.



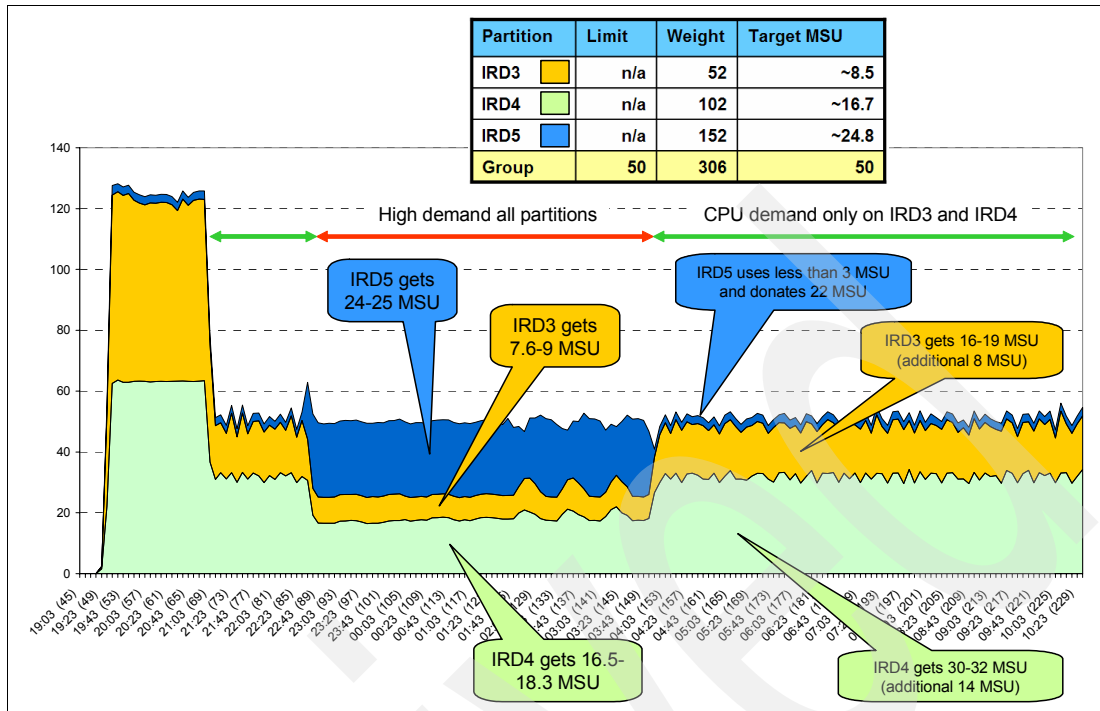


Figure 10-16 Group capacity limit usage for example 2

In Figure 10-17, you see the same LPARs with 4-hour rolling average values within the same interval as in Figure 10-16. The IPL bonus that is given can be seen significantly in this example.

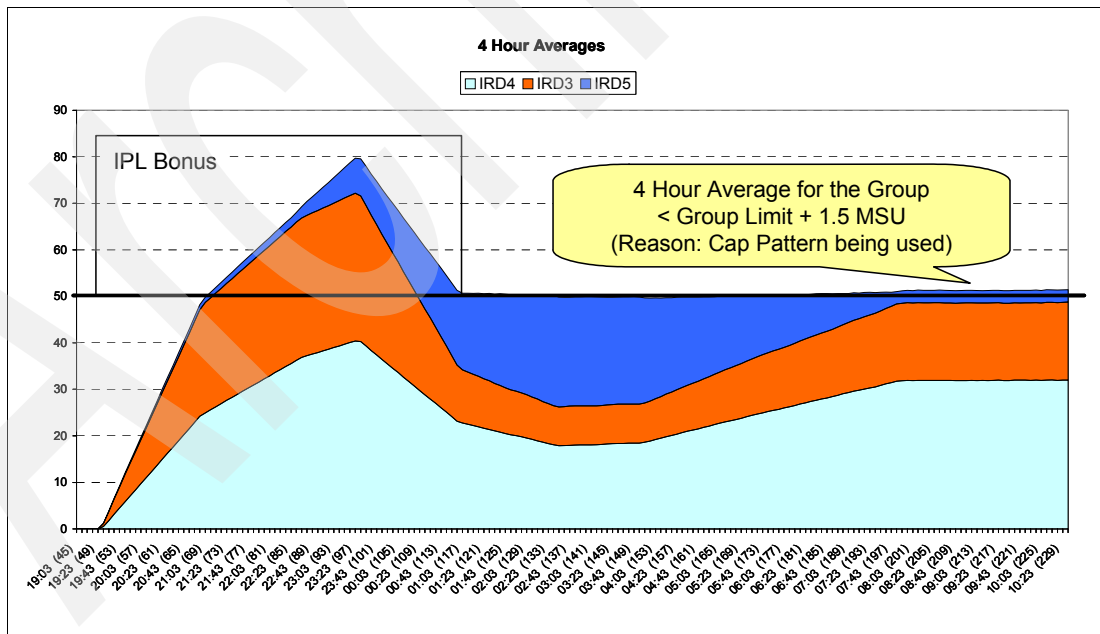


Figure 10-17 Group capacity limit usage example with the 4-hour rolling average

In Figure 10-18 on page 153, LPAR IRD3 capping and unused capacity values are shown, as well as the actual MSU values for both the capped and uncapped portions.



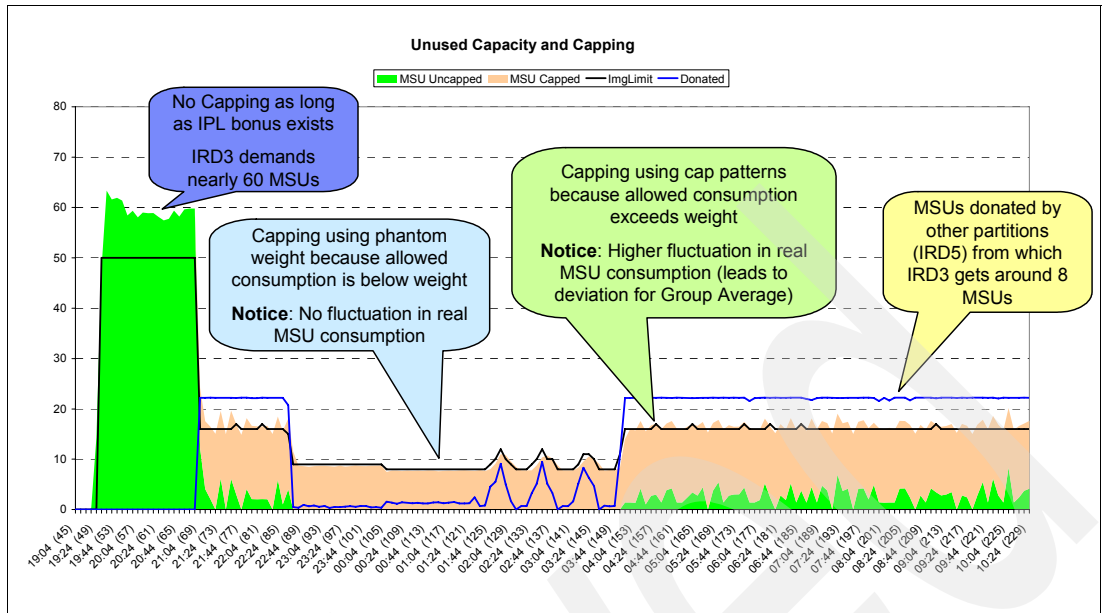


Figure 10-18 Group capacity limit usage for example 2 with LPAR IRD3 in detail

Archived

## C/C ++ enhancements

The C language is a general purpose, versatile, and functional programming language that allows a programmer to create applications quickly and easily. C provides high-level control statements and data types as do other structured programming languages. It also provides many of the benefits of a low level language.

The C++ language is based on the C language and includes all of the advantages of C. In addition, C++ also supports object-oriented concepts, generic types or templates, and an extensive library. The C++ language introduces classes, which are user-defined data types that may contain data definitions and function definitions. You can use classes from established class libraries, develop your own classes, or derive new classes from existing classes by adding data descriptions and functions. New classes can inherit properties from one or more classes. Not only do classes describe the data types and functions available, but they can also hide (encapsulate) the implementation details from user programs. An *object* is an instance of a class. The C++ language also provides templates and other features that include access control to data and functions, and better type checking and exception handling. It also supports polymorphism and the overloading of operators.

This chapter describes:

- ▶ SUSv3 implementation in z/OS V1R9
- ▶ Compiling an SUSv3 application
- ▶ Invoking thread support
- ▶ Setting environment variables affects run-time behavior
- ▶ New APIs
- ▶ New Threading Interfaces
- ▶ Modified APIs
- ▶ Software Dependencies
- ▶ Migration and coexistence considerations
- ▶ Language Environment C run-time is missing SUSv3 support

## 11.1 SUSv3 implementation in z/OS V1R9

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.

Language Environment's implementation of SUSv3 is based on IEEE Std 1003.1-2004, which is comprised of IEEE STD 1003.1-2001 and the two subsequent corrigenda issued after the initial release of the new standard.

From the C/C++ and LE point of view, many standards have been implemented in the latest z/OS releases and we continue with this implementation in z/OS V1R9.

**Note:** Language Environment C run-time APIs are compatible with most of SUSv3, although not fully compliant.

Today, C applications use APIs and expect behaviors that were not implemented in the Language Environment C run-time prior to z/OS V1R9. z/OS V1R9 has updated the XL C/C++ run-time Library with changes designed to meet the X/Open System Interface extension as defined in the Single Unix Specification, Version 3. z/OS V1R9 has enhancements to make it easier to develop, port, and deploy contemporary C/C++ applications on z/OS in the following areas:

- ▶ Support for the thread option (pthread interfaces)
- ▶ Addition of any other missing headers, APIs, and functionality

### 11.1.1 z/OS V1R9 and SUSv3

The SUSv3 implementation adds a number of new functions to the XL C/C++ Run-Time Library, while applying modifications to the signatures of some existing functions in SUSv3 as result of the following behaviors:

- ▶ Addition or removal of arguments
- ▶ Use of const declarator
- ▶ Specialization of argument types
- ▶ Use of restrict keyword

The SUSv3 namespace excludes all withdrawn headers, functions, external variables, and constants. The specification further targets additional symbols for removal in a future version of the standard.

**Note:** Applications compiled for SUSv3 are also implicitly C99, and therefore there is no need to define feature test macro `_ISOC99_SOURCE` when `_XOPEN_SOURCE 600` or `_POSIX_C_SOURCE 200112L` is defined.

Another behavioral difference is the change in the return value of most threads functions. The POSIX.4a, draft 6 threads behavior indicates a return of -1 on failure with the error code set in `errno`. In SUSv3, the majority of these functions now return the error code on failure rather than a value of -1. With the exception of `pthread_getspecific()`, the z/OS implementation will continue to set `errno` in addition to returning the error code.

## Feature test macros

Many of the symbols that are defined in headers are “protected” by a feature test macro (FTM). These protected symbols are invisible to the application unless the user defines the feature test macro with #define, using either of the following methods:

- ▶ In the source code before including any header files.
- ▶ On the compilation command

With z/OS V1R9, the following feature test macros are new:

- ▶ `_IEEEV1_COMPATIB`
- ▶ `__STDC_WANT_DEC_FP__`
- ▶ `_UNIX03_THREADS`
- ▶ `_UNIX03_WITHDRAWN`

**Note:** The `LANGLVL` compiler option does not define or undefine these macros.

### 11.1.2 Compiling an SUSv3 application

C++ applications can access C99 run-time library functions by using feature test macros. To expose C99 interfaces, C++ applications can define the appropriate feature test macros before including the identified header.

Single UNIX Specification, Version 3 aligns with ISO/IEC 9899:1999, and is commonly referred to as the “C99 language standard”. In some cases, SUSv3 extends the C99 definition, although in the case of a conflict, it always defers to the C99 standard. For this reason, applications compiled for SUSv3 are also implicitly C99, and you do not need to define feature test macro `_ISOC99_SOURCE` when `_XOPEN_SOURCE 600` or `_POSIX_C_SOURCE 200112L` is defined. Some applications define feature test macros to inform the compiler and to expose a namespace, as follows:

- ▶ `_POSIX_C_SOURCE 200112L`

This defines the SUSv3 POSIX symbols and prototypes, and incorporates the namespaces defined by `_POSIX_SOURCE`, `_POSIX1_SOURCE`, `_MSE_PROTOS`, and `_ISOC99_SOURCE`. These feature test macros are redundant in this context and do not need to be defined separately.

- ▶ `_XOPEN_SOURCE 600`

This defines the X/Open System Interface (XSI) symbols and prototypes, and incorporates all the namespace of `_POSIX_C_SOURCE 200112L` as well as those of `_UNIX02+THREADS`, `_OPEN_SYS_MUTEX_EXT`, and `_LARGE_FILE` (if compiling with `langlvl(longlong)` or its equivalent). These feature test macros are redundant in this context and do not need to be defined separately.

#### **`_UNIX03_SOURCE`**

This feature test macro exposes new Single UNIX Specification, Version 3 interfaces. It does not change the behavior of existing APIs, nor expose interfaces controlled by feature test macros such as `_XOPEN_SOURCE_EXTENDED`. Functions and behavior exposed by `_UNIX03_SOURCE` are a subset and not the complete implementation of the Single UNIX Specification, Version 3.

It defines the functions that were added prior to z/OS V1R9. The following functions have been added under `_UNIX03_SOURCE`:

- ▶ z/OS V1R06: `dlclose()`, `dlerror()`, `deopen()`, and `dlsym()`
- ▶ z/OS V1R07: `sched_yield()`, `strerror_r()`, and `unsetenv()`
- ▶ z/OS V1R08: `flockfile()`, `ftrylockfile()`, `funlockfile()`, `getc_unlocked()`, `getchar_unlocked()`, `putc_unlocked()`, and `putchar_unlocked()`
- ▶ z/OS V1R09: `posix_openpt()`, `pselect()`, and `socketatmark()`

### Functions withdraw in SUSv3

If it is necessary to continue using the following functions in an application written for Single UNIX Specification, Version 3, then define the feature test macro `_UNIX03_WITHDRAWN` before including any standard system headers. The macro exposes all interfaces and symbols removed in Single UNIX Specification, Version 3.

Following are the functions, headers, and constants withdrawn from SVSv3:

- ▶ Functions: `advance()`, `brk()`, `chroot()`, `compile()`, `cuserid()`, `gamma()`, `getdtablesize()`, `getpagesize()`, `getpass()`, `getw()`, `putw()`, `re_comp()`, `re_exec()`, `regcmp()`, `regex()`, `sbrk()`, `sigstack()`, `step()`, `ttyslot()`, and `valloc()`
- ▶ Headers: `<re_comp.h>`, `<regexp.h>`, and `<varargs.h>`
- ▶ Constants: `loc1`, `_loc1`, `loc2`, `locs`, `L_cuserid`, `NOSTR`, and `YESSTR`

## 11.1.3 Invoking Threads support

Several other implementation-specific feature test macros impact support of SUSv3. An application may request SUSv3 threads support by defining `_UNIX03_THREADS` or as part of XSI. Because the Threads Option is a required component of XSI, you do not need to define `_UNIX03_THREADS` when `_XOPEN_SOURCE 600` is defined. Another new macro, `_UNIX03_WITHDRAWN`, preserves symbols withdrawn from the UNIX standard, making them visible in the SUSv3 namespace.

**Note:** Program developers have the option of writing SUSv3 applications that still use the old threads behavior. An application may override the implicit XSI threads behavior by defining both `_OPEN_THREADS` and `_XOPEN_SOURCE 600`, if there is a reason to maintain the previous POSIX.4a, draft 6 behavior.

On the other hand, concurrent definition of the `_UNIX03_THREADS` and `_OPEN_THREADS` macros is not allowed and will generate a compile-time error message.

Language Environment provides additional support in the C/C++ run-time library for SUSv3, including new APIs and behaviors for the Thread functions, as follows:

- ▶ `_UNIX03_THREADS`  
It provides support for SUSv3 threads option. The same support is provided if you define `_XOPEN_SOURCE 600`, unless there is an override.
- ▶ `_OPEN_THREADS`  
It supports the POSIX.4a, draft 6 threading model. It is mutually exclusive with `#define _UNIX03_THREADS`.

► `_OPEN_THREADS 2`

Adds support to SUSv3 pthread APIs introduced in z/OS V1R7 to POSIX.4a base.

- Functions: `pthread_getconcurrency()`, `pthread_setconcurrency()`, `pthread_setcancelstate()`, `pthread_setcanceltype()`, `pthread_sigmask()`, `pthread_testcancel()`, `pthread_key_delete()`
- Constants: `PTHREAD_CANCEL_ENABLE`, `PTHREAD_CANCEL_DISABLE`, `PTHREAD_CANCEL_DEFERRED`, `PTHREAD_CANCEL_ASYNCHRONOUS`

► `_OPEN_THREADS 3`

Adds support to SUSv3 pthread APIs introduced in z/OS V1R7 and z/OS V1R9 to POSIX.4a base.

- Functions: `pthread_atfork()`, `pthread_attr_getguardsize()`, `pthread_attr_getschedparam()`, `pthread_attr_getstack()`, `pthread_attr_getstackaddr()`, `pthread_attr_setguardsize()`, `pthread_attr_setschedparam()`, `pthread_attr_setstack()`, `pthread_attr_setstackaddr()`
- Constants: `PTHREAD_CANCEL`, `PTHREAD_COND_INITIALIZER`, `PTHREAD_CREATE_DETACHED`, `PTHREAD_CREATE_JOINABLE`, `PTHREAD_EXPLICIT_SCHED`

**Note 1:** The feature test macros `_UNIX03_THREADS` and `_OPEN_THREADS` may not be used together. If both are defined, the `features.h` header will generate an error during compile time.

**Note 2:** Defining `_XOPEN_SOURCE 600` assumes `_UNIX03_THREADS` support for the application, but a user may choose to override the pthread behavior by defining `_OPEN_THREADS` with one of its allowed values. This combination allows the possibility of getting all the new SUSv3 support (except for threads) while maintaining POSIX.4a draft6 behavior in an existing application.

**Note 3:** Either `_UNIX03_THREADS` or `_XOPEN_SOURCE 600` establish a complete SUSv3 implementation of the Threads option.

**Note 4:** One major difference between SUSv3 thread interfaces and the POSIX.4a Draft 6 implementation concerns failures. Most of the SUSv3 versions return an error value if they do not complete successfully, whereas the Draft 6 versions return -1 and set `errno`. The Language Environment implementation of SUSv3 continues to set `errno` in these functions in addition to returning the `errno` value on failure.

**Note 5:** SUSv3 adds support for static initialization of condition variables.

### 11.1.4 Setting environment variables affects run-time behavior

The following z/OS XL C/C++ specific environment variables are supported to provide various functions. z/OS XL C/C++ variables have the prefix `_CEE_` or `_EDC_`. Do *not* use these prefixes to name your own variables. The following environment variables affect run-time behavior:

► `_EDC_SUSV3`

This indicates 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 will not check for these conditions. When the value of `_EDC_SUSV3` is set to 1, the SUSV3 behavior is enabled.

Existing programs using these interfaces and running with `_EDC_SUSV3=1` might fail in z/OS V1R9 with an `errno`, where in the past, they might have appeared to succeed. The user must set `_EDC_SUSV3=1` to get the new behavior.

z/OS XL C/C++ Run-Time Library Reference documents the use of `_EDC_SUSV3` in individual interface descriptions. The functions that are affected by the `_EDC_SUSV3` environment variable are: `setenv()`, `readdir()`, `getnameinfo()`, and `tcgetsid()`.

The `_EDC_SUSV3` environment variable can be set with the function:

```
setenv("_EDC_SUSV3", "1", 1)
```

Value	Description
1	Enable SUSv3 behavior for <code>setenv()</code> , <code>readdir()</code> , <code>getnameinfo()</code> , and <code>tcgetsid()</code> .
0	Functions perform without SUSV3 behavior. This is the default. It is equivalent to unsetting the environment variable.

► `_EDC_EOVERFLOW`

This sets the behavior of the `ftell()`, `fseek()`, `fstat()`, `lstat()`, `stat()`, and `mmap()` functions. By default, these functions will not check for the `Eoverflow` error condition. Setting `_EDC_EOVERFLOW` to `YES` enables testing for this condition, and if overflow is detected, setting `errno` to `Eoverflow` and returning an error.

The `_EDC_EOVERFLOW` environment variable can be set with the function:

```
setenv("_EDC_EOVERFLOW", "YES", 1)
```

Value	Description
YES	Check for <code>Eoverflow</code> error conditions.
<other>	Ignore setting of <code>Eoverflow</code> . This is the default. It is equivalent to unsetting the environment variable.

`Eoverflow` indicates a file offset or some other file attribute too large to represent in its container (that is, the datatype is too small). It is not possible to reach this error in a 64-bit application. `_EDC_EOVERFLOW` is provided as a way to give users the option to turn off the new behavior in applications in which the affected functions may have appeared successful and were not setting the `Eoverflow` `errno`.

**Note:** The guideline for integrating new behavior into existing library code was that recompilation of an existing application could not lead to different results with respect to success or failure on the call to the affected function. This class of change was seen as an extension rather than needing a separate SUSv3 interface. Functions not meeting this criterion were introduced in a separate interface, or isolated with an environment variable.

## 11.1.5 New APIs

The following new functions are introduced with SUSv3:

► **posix\_openpt()** - Open a pseudo-terminal device

The `posix_openpt()` function establishes a connection between a master device for a pseudo-terminal and a file descriptor. The file descriptor is used by other I/O functions that refer to that pseudo-terminal. This complements existing functions: `grantpt()`, `ptsname()`, `unlockpt()` associates a file descriptor with a master device for a given pseudo-terminal

► **pselect()** - Monitor activity on files/sockets/message queues

The `pselect()` and `select()` functions monitor activity on a set of sockets and/or a set of message queue identifiers until a timeout occurs, to see if any of the sockets and message



queues have read, write, or exception processing conditions pending. This call also works with regular file descriptors, pipes, and terminals. This is very similar to `select()`, except that `pselect()` supports timeout expressions using nanoseconds `sigmask` as an argument modification of the timeout object upon successful completion. `pselect()` is not affected by `_OPEN_MSGQ_EXT` or `_OPEN_SYS_HIGH_DESCRIPTOR`s.

- ▶ **socketatmark()** - Determine if socket is at the out-of-band data mark

The `socketatmark()` function determines whether the socket specified by the descriptor `s` is at the out-of-band data mark. If the protocol for the socket supports out-of-band data by marking the stream with an out-of-band data mark, the `socketatmark()` function returns 1 when all data preceding the mark has been read and the out-of-band data mark is the first element in the receive queue. The `socketatmark()` function does not remove the mark from the stream. If the protocol for the specified socket supports marking out-of-band data, then `socketatmark()` returns one of the following:

- 1 Stream has been marked and all data preceding the mark has been read.
- 0 No mark, or data in the receive queue still precedes the mark.
- 1 Error; `errno` is set.

- ▶ **netinet/tcp.h** - Definitions for the Internet Transmission Control Protocol (TCP)

This is a new header, first released in Issue 6. It is derived from the XNS, Issue 5.2 specification.

- ▶ **sys/select.h** - Type definitions for `select()` / `pselect()`

This is a new header, first released in Issue 6. It is derived from IEEE Std 1003.1g-2000.

## 11.1.6 New Threading interfaces

Several Thread attribute accessors are added:

- ▶ `pthread_attr_getguardsize()`, `pthread_attr_setguardsize()`

`pthread_attr_getguardsize()` gets the `guardsize` attribute from `attr` and stores it into `guardsize`. `attr` is a pointer to a thread attribute object initialized by `pthread_attr_init()`. The retrieved `guardsize` always matches the size stored by `pthread_attr_setguardsize()`, despite internal adjustments for rounding to multiples of the `PAGESIZE` system variable.

- ▶ `pthread_attr_getstack()`, `pthread_attr_setstack()`

The `pthread_attr_getstack()` function gets both the base (lowest addressable) storage address and size of the initial stack segment from a thread attribute structure and stores them into `addr` and `size` respectively. `attr` is a pointer to a thread attribute object initialized by `pthread_attr_init()`. Get and set the thread stack size and stack address attributes. It is used for application-specified stack. It is not recommended on z/OS, but is available for compliance. It is better to let Language Environment handle the stack.

- ▶ `pthread_attr_getstackaddr()`, `pthread_attr_setstackaddr()`

The `pthread_attr_getstackaddr()` function gets the `stackaddr` attribute from `attr` and stores it into `addr`. The `stackaddr` attribute holds the storage location of the created thread's initial stack segment. `attr` is a pointer to a thread attribute object initialized by `pthread_attr_init()`. Get and set the thread stack address attribute. It is also used for application-specified stack. It is not recommended on z/OS (or in the UNIX standard), but required for compliance, even though marked obsolescent in the SUSv3 standard.

- ▶ `pthread_attr_getschedparam()`, `pthread_attr_setschedparam()`

`pthread_attr_getschedparam()` gets the scheduling priority attribute from `attr` and stores it into `param`. `attr` is a pointer to a thread attribute object initialized by `pthread_attr_init()`.

param points to a user-defined scheduling parameter object into which pthread\_attr\_getschedparam() copies the thread scheduling priority attribute. Get and set the thread schedparam attribute. Being of minimal implementation, it is included for compliance. z/OS services always control scheduling on the platform. It is an implemented the interface for registering and executing fork handlers.

▶ pthread\_atfork()

The pthread\_atfork() function registers fork handlers to be called before and after fork(), in the context of the thread that called fork(). Fork handler functions may be named for execution at the following three points in thread processing:

- The prepare handler is called before fork() processing commences.
- The parent handler is called after fork() processing completes in the parent process.
- The child handler is called after fork() processing completes in the child process.

This function registers 3-tuples of fork handlers (prepare, parent, and child) to be run immediately before the fork and in the child and parent after a successful fork. Not all three need to be present.

### 11.1.7 Modified APIs

The following APIs are modified in z/OS V1R9;

- ▶ sysconf() now supports required SUSv3 constants.
- ▶ confstr() now supports required SUSv3 constants.
- ▶ getnameinfo() is enhanced to use NI\_NUMERICSCOPE and EAI\_OVERFLOW.
- ▶ gai\_strerror() is extended to support EAI\_OVERFLOW.
- ▶ sigwait() prototype adds a second arg to return the signal ID.
- ▶ getdate() handles a new %C format specifier for century number.

### 11.1.8 Migration and coexistence considerations

The z/OS C/C++ compiler must be C99-compliant if SUSV3 applications use features of C99 that require this compiler level (for example, complex.h functionality).

Set environment variable `_EDC_OVERFLOW=YES` to allow EOVERFLOW reporting in ftell(), fseek(), fstat(), lstat(), stat(), and mmap()

Set environment variable `_EDC_SUSV3=1` to get new SUSv3 error handling for setenv(), readdir(), getnameinfo(), and tcgetsid().

Compile with the `_UNIX03_WITHDRAWN` feature test defined to preserve visibility of elements removed from SUSv3, such as Legacy Feature Group from SUSv2.

Review differences in SUSv3 function signatures and behavior.

**Note:** Default settings of `_EDC_OVERFLOW` and `_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.



## ISPF enhancements

This chapter describes the enhancements to ISPF in z/OS V1R9. The enhancements in z/OS V1R9 concentrate mostly on cross-platform support.

The following topics are discussed:

- ▶ Edit and browse z/OS UNIX files
- ▶ Support for editing ASCII files
- ▶ Mixed case characters in ISPF command tables

## 12.1 Edit and browse z/OS UNIX files

In z/OS V1R8, ISPF was enhanced to process z/OS UNIX files. The z/OS V1R8 support includes the ability to edit, browse, create, delete, rename, copy and replace z/OS UNIX files. This support is implemented as a directory list utility. The utility is available as option 17, z/OS UNIX Directory List Utility, under the ISPF Utilities menu (option 3 from the main ISPF panel).

The z/OS UNIX Directory List Utility provides a subset of the functions supported by ISHELL. The functions supported by the directory list utility are aimed to assist with basic tasks undertaken by ISPF users such as programmers, and not z/OS UNIX administrators. The z/OS UNIX Directory List Utility was designed to behave and support commands similar to the data set list utility (ISPF option 3.4). The intention is to assist users already familiar with ISPF functions to work with z/OS UNIX files in an interface similar to what they are already used to, instead of OMVS or ISHELL.

In z/OS V1R8, however, the OEDIT and OBROWSE commands, used to edit and browse z/OS UNIX files, use the EDIF and BRIF services to interface with the ISPF EDIT and BROWSE functions. This does not provide the full functionality available with the native ISPF EDIT and BROWSE functions.

### 12.1.1 ISPF enhancement in z/OS V1R9

In z/OS V1R9, the ISPF EDIT, VIEW and BROWSE functions are enhanced to support z/OS UNIX files. This includes support for z/OS UNIX files via options 1 and 2 on the main ISPF panel, as well as the ISPF EDIT, VIEW and BROWSE service interfaces. In addition, the Edit Entry, Browse/View Entry, Edit Copy, Edit Move and Edit Create panels are modified so that the “Other” data set name field is a scrollable field which supports z/OS UNIX files and path names up to 1023 characters in length. ISPF assumes a z/OS UNIX path name is entered in this field when the first character is one of the following:

<b>/ (forward slash)</b>	Identifies an absolute path name
<b>~ (tilde)</b>	Represents the path name for the user’s home directory.
<b>. (dot)</b>	Represents the path name for the current working directory.
<b>.. (dot dot)</b>	Represents the path name for the parent directory of the current working directory.

Following are examples of the use of these characters:

- ▶ Assuming a user’s home directory is /u/smith, specifying ~/test/tst1.sh is the equivalent of specifying the absolute path name /u/smith/test/tst1.sh.
- ▶ Specifying ./pgma.c is the equivalent of specifying the absolute path name /u/proj1/dev/pgma.c when the current working directory is /u/proj1/dev/.
- ▶ Specifying ../test/pgma.c is the equivalent of specifying the absolute path name /u/proj1/test/pgma.c when the current working directory is /u/proj1/dev/.

#### ISPF Edit Entry Panel

Figure 12-1 on page 165 displays the new look of the “Other” data set name field on the Edit panels, as follows:

Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:

As shown in Figure 12-1 on page 165, the new Edit Entry Panel and View Entry Panel now support z/OS UNIX file names on the “Other” data set name field.

## Record length field

Another new field on these panels is the Record length field. The Record length field is added to allow a record length to be specified when browsing a z/OS UNIX file. The value entered in this field is used by ISPF to display the data in the file as fixed-length records, rather than using the new line character to delimit each record. This is useful for browsing files which would otherwise have very long records if the new line character is used as the record delimiter.

```
Menu RefList RefMode Utilities Workstation Help
-----
Edit Entry Panel
Command ==> _____
ISPF Library:
Project . . . . PELEG
Group . . . . JCL
Type . . . . VERS5
Member . . . . _____ (Blank or pattern for member selection list)
Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
Name . . . . /u/peleg
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 . . . . _____
```

Figure 12-1 z/OS UNIX files support on the Edit Entry Panel

**Note:** The Record Length field can be used when editing a z/OS UNIX file. It allows you to specify a record length which is used by the editor to load the records from the file into the edit session as fixed-length records. When the file is saved, it is saved with fixed length records. So the Record Length field allows you to convert a variable length file to a fixed length file.

The value specified in the Record Length field must be able to accommodate the largest record in the file. If the editor finds a record that is larger than the length specified, an error message is displayed and the edit session does not proceed.

## Browsing z/OS UNIX files

On the Edit Entry Panel, the Record length field has a slightly different affect than on the View Entry Panel. With EDIT, ISPF normally treats z/OS UNIX files as having variable length records. The Record length field on the Edit Entry Panel allows you to specify a record length which ISPF EDIT will use to load the records from the file into the edit session as fixed length records. When the file is saved, it is saved with fixed length records. So the Record length field allows you to convert a variable length records file into a fixed length records file.

Figure 12-2 on page 166 and Figure 12-3 on page 166 show the difference when browsing a z/OS UNIX file without specifying a record length and when browsing a z/OS UNIX file with record length of 40 specified. The z/OS UNIX file that is browsed is:

```
/Z19RB1/samples/magic
```

```

Menu Utilities Compilers Help
-----
BROWSE      /Z19RB1/samples/magic                               Line 00000000 Col 001 072
Command ==> _ Scroll ==> HALF
***** Top of Data *****
# Magic
# .Magic file for "file" command.
#
# .The fields of this file are as follows:
# .byte offset, value type, optional operator (= by default), value
# .to match (numeric or string), and string to be printed. Numeric
# .values may be decimal, octal, or hex. Also note that the last
# .string may have 1 printf format spec.
# .The '>' in occasional column 1's is magic: it forces file to
# .continue scanning and matching additional lines. The first line
# .afterwards not so marked terminates the search.
#
# .Magic numbers are the byte codes at the start of a file which indicate

```

Figure 12-2 Browsing a z/OS UNIX file, no record length specified

Figure 12-3 illustrates browsing a z/OS UNIX file with record length of 40 specified.

```

Menu Utilities Compilers Help
-----
BROWSE      /Z19RB1/samples/magic                               Line 00000000 Col 001 040
Command ==> _ Scroll ==> HALF
***** Top of Data *****
# Magic.# .Magic file for "file" command
.#.#.#.The fields of this file are as fol
lows:#.byte offset, value type, optiona
l operator (= by default), value.#.to ma
tch (numeric or string), and string to b
e printed. Numeric.#.values may be deci
mal, octal, or hex. Also note that the
last.#.string may have 1 printf format s
pec.#.The '>' in occasional column 1's
is magic: it forces file to.#.continue s
canning and matching additional lines.

```

Figure 12-3 Browsing a z/OS UNIX file, record length of 40 specified

### ISPF Edit Entry Panel usage

From Figure 12-1 on page 165 with the z/OS UNIX file specified (/u/peleg), when the Enter key is used, the z/OS UNIX Directory List utility is invoked by ISPF to display the contents of the directory, as shown in Figure 12-4 on page 167.

Figure 12-4 on page 167 now shows that previously file names, aaaa, bapi.c and mylibar.c, have been edited by specifying a record length. This is seen in the Fmat field with the specification of the file format shown as n1 (new line).

```

Menu Utilities View Options Help
-----
z/OS UNIX Directory List Row 1 to 21 of 25
Command ==> _____ Scroll ==> PAGE
Pathname . : /u/peleg

Command  Filename      Message      Type Permission Audit  Ext  Fmat
-----
_____ .                Dir  rwxr-xr-x  fff--
_____ ..               Dir  r-xr-xr-x  -----
_____ .profile          File rwx----- fff-- --s- ----
_____ .sh_history       File rw----- fff-- --s- ----
_____ a.out             File rwxr-xr-x fff-- --s- ----
_____ aaaa             File rwx----- fff-- --s- nl
_____ bapi             File rwxr-xr-x fff-- --s- ----
_____ bapi.c           File rwx----- fff-- --s- nl
_____ bapi.o           File rw-r--r-- fff-- --s- ----
_____ bapi2            File rwxr-xr-x fff-- --s- ----
_____ bapi2.c          File rwx----- fff-- --s- ----
_____ bapi2.o          File rw-r--r-- fff-- --s- ----
_____ mylibar.a        File rw-r--r-- fff-- --s- ----
_____ mylibar.c        File rwx----- fff-- --s- nl
_____ mylibar.o        File rw-r--r-- fff-- --s- ----
_____ mulibar.x        File rw-r--r-- fff-- --s- ----

```

Figure 12-4 The z/OS UNIX Directory List panel

**Note:** Beginning with z/OS V1R8, z/OS UNIX files could have a file format. The possible file formats are listed in Table 12-1 on page 167.

Table 12-1 Possible file formats for z/OS UNIX files

Value	Format options
NA	not specified
BIN	binary data file
Value	Text data delimiters
NL	new line
CR	carriage return
LF	line feed
CRLF	Carriage Return followed by Line Feed
LF CR	Line Feed followed by Carriage Return
CR NL	Carriage Return followed by New Line

### ISPF ENQ requests on z/OS UNIX files

If you enter a forward slash (/) in the Command field next to one of the file names, the following Directory List Actions panel pops up, as shown in Figure 12-5 on page 168.

To avoid possible data corruption, ISPF issues an exclusive ENQ request to prevent two or more users from editing the same z/OS UNIX file at the same time. The ENQ major name is SPFEDIT. The ENQ minor name is a 12-byte string comprising these three binary integers:

- ▶ The file's i-node number (4 bytes)
- ▶ The file's device number (4 bytes)
- ▶ A sysplex indicator (4 bytes) is set to 1 when z/OS UNIX is running with SYSPLEX(YES)

This is the same ENQ that is issued by the **0EDIT** command.

```
Directory List Actions

File ----- /u/peleg/bapi.c

DIRLIST Action
— 1. Edit           8. Copy Out
  2. View          9. Copy In
  3. Browse       10. Information
  4. New          11. Modify Mode Fields
  5. Directory List 12. Modify Extended Attrs
  6. Delete       13. Execute command
  7. Rename

Select a choice and press ENTER to process data set action.

F1=Help   F3=Exit   F12=Cancel
```

Figure 12-5 Directory List Actions panel

## 12.2 ISPF personal data set lists

ISPF personal data set lists are enhanced in z/OS V1R9 to support saving and retrieving z/OS UNIX file path names. Retrieval of path names is supported on the ISPF Edit, View, and Browse panels to allow a path name to be entered in the “Other” data set name field. Retrieval of path names is also supported on the z/OS UNIX Directory List Entry panel. Users can also add path names manually to their personal data set lists.

Personal data set lists are a good way to group (by project, for example) data sets and z/OS UNIX file path names that you use frequently. You can use personal data set lists to avoid typing in data set names and z/OS UNIX file path names, and to create customized data set lists similar to those using ISPF Option 3.4.

Figure 12-6 on page 169 shows an example of an ISPF personal data set list with z/OS UNIX file names mixed together with MVS data sets.

### REFACTD command

The **REFACTD** command can be used to display the same Personal Data Set List panel. If you have one or more personal data set lists, ISPF displays the current list. If you have no personal data set lists, ISPF displays the reference list called REFLIST, which is updated by ISPF whenever a new data set is used by ISPF.



```

File View Options Help
-----
Personal Data Set List
Command ==> _
Enter a list action to perform or select a data set entry to retrieve.
Action: S=Save A=Save As D=Delete this list E=Extended Edit L=DSLISL

Action Name Description Created Referenced
_ REFLIST Last 30 referenced data sets 07/08/01 15:08
More: +
Select Data Set, DSLISL Level or z/OS UNIX file Volume WS
. /u/peleg
. /u/peleg/temp
. /u/rogers
. /u/rogers/rich.txt
. /u/rogers/tst102
. /u/rogers/CEEDUMP.2006061
. /u/rogers/CEEDUMP.20060614.094113.67305623
. /Z19RB1/sampLes/magic
. 'SYS1.TCPPARMS'
. 'CPAC.PROCLIB'
. 'SYS1.IBM.PROCLIB'
. 'SYS1.PARMLIB'
. 'HAIMO.OUTPUT'
. 'HAIMO.RESOLV.OMP'
. 'SYS1.PROCLIB'
. 'SYS1.PROCLIB' TCBSY1
. 'SYS1.PARMLIB' TCBSY1
. QESVP.EXEC

```

Figure 12-6 ISPF personal data set list with z/OS UNIX file names

## 12.3 EDIT primary commands support

While you are using the PDF editor to edit or view data, the following primary commands can be entered on the command line, as they now support the specification of a z/OS UNIX path name as an operand with z/OS v1R9:

- ▶ COMPARE command
- ▶ COPY command
- ▶ CREATE command
- ▶ MOVE command
- ▶ REPLACE command

The **pathname** is new and is specified in the same format accepted by the “Other” data set name field on the edit panels. If these commands are used when editing a z/OS UNIX file, the “|” character can be specified as the first character of the path name to represent the directory name of the directory containing the file currently being edited.

**Note:** For more information about the syntax of these commands, refer to *z/OS ISPF Edit and Edit Macros*, GC34-4820.

### The COMPARE command

The **COMPARE** command can now be used to compare a member, data set or z/OS UNIX file being edited with another member, data set, or z/OS UNIX file. The new syntax of the **COMPARE** command is shown in Figure 12-7 on page 170, where **dsname** is the name of a member, data set, or z/OS UNIX file to which the current file is compared.

```

>>--COMPARE--| -dsname-----| --|-----| --|-----| --|-----|-----><
              | -NEXT-----|  | -EXCLUDE-|  | -SAVE-|  | -SYSIN-|
              | -| -SESSION--|
              | - * -----|
              | - / -----|

```

Figure 12-7 Syntax of the COMPARE command

Figure 12-8 displays an ISPF edit screen for file /u/rogers/rich.txt. You can see, on the Command line, the command `compare /u/peleg/aaaa`. When the Enter key is pressed, file /u/rogers/rich.txt is compared to file /u/peleg/aaaa.

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT      /u/rogers/rich.txt          Columns 00001 00072
Command ==> compare /u/peleg/aaaa_   Scroll ==> HALF
***** ***** Top of Data *****
000001 asdfasdf
000002 asdfasdfasdf
***** ***** Bottom of Data *****

```

Figure 12-8 Using the COMPARE command to compare to z/OS UNIX files

If the file pathname specified to be compared against is larger than the number of characters that can be entered on the Command line, then specify the `dsname /` to indicate a long path name is required. A pop-up window containing a scrollable field for entering the long path name is displayed, as shown in Figure 12-9.

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
RefList RefMode
Edit Compare - z/OS UNIX File Entry
Command ==>
Pathname . . . /_
Instructions:
Provide a z/OS UNIX file pathname and press ENTER to proceed with the
COMPARE.
Enter END , EXIT or CANCEL to cancel the COMPARE.
F1=Help F3=Exit F4=Expand F12=Cancel

```

Figure 12-9 Specify a long path name for the COMPARE command

### The COPY command

The `COPY` command can now be used to copy data from a z/OS UNIX file into a member, data set or z/OS UNIX file currently being edited. To do this, specify the path name for the z/OS UNIX file as a parameter with the `COPY` command. You can also specify the path name on the Edit/View Copy panel that is displayed when no data source is specified with the `COPY` command. If the path name for a directory is specified, the z/OS UNIX Directory List utility is invoked to allow you to select the regular file to be copied.

The new syntax of the `COPY` command is shown in Figure 12-10 on page 171.

```

>>--COPY--|-----|---|-----|---|label|-----><
          |-member-----|   |---|-----|---|
          |-(member)-----|   |---|-----|---|
          |-dsname-----|   |---|-----|---|
          |-dsname(member)-|   |---|-----|---|
          |-pathname-----|   |---|-----|---|
>-----<
          |-start_line--end_line-|

```

Figure 12-10 Syntax of the COPY command

For example, if editing file /u/usr1/prog1 entering on the command line, `copy /u/usr1/src1`, copies data from file src1 into the current file being edited.

### The CREATE command

The **CREATE** command can now be used to create a z/OS UNIX regular file from the data currently being edited. If currently editing a z/OS UNIX file and the **CREATE** command is used to create a new z/OS UNIX file, then the file permissions for the new file are set to the same values as the file permissions of the file being edited. If currently editing a sequential data set or a member and the **CREATE** command is used to create a new z/OS UNIX file, then the file permissions are set to 700 (rwx --- ---).

The new syntax of the **CREATE** command is shown in Figure 12-11.

```

>>---CREATE-----><
  |-CRE-----|   |-member-----|   |-labela--labelb-|
                |-(member)-----|
                |-dsname(member)-|
                |-dsname-----|
                |-pathname-----|

```

Figure 12-11 Syntax of the CREATE command

Where pathname is used to specify the path name of a regular z/OS UNIX file to be created.

In Figure 12-12, the **CREATE** command is used to create a member in the file system. PDS member ROGERS.JCL.VERS5(GENER) is created as file GENER in directory /u/rogers.

```

File Edit Edit_Settings Menu Utilities Compilers Iest Help
-----
EDIT          ROGERS.JCL.VERS5(GENER) - 01.15          Columns 00001 00072
Command ==> create /u/rogers/gener          Scroll ==> HALF
***** ***** Top of Data *****
CC0100 //ROGERSC JOB (POK,999),MSGCLASS=T,NOTIFY=ROGERS
000200 //PRINT EXEC PGM=IEBGENER
000300 //SYSPRINT DD SYSOUT=T
000400 //SYSUT2 DD SYSOUT=J
000500 //SYSUT1 DD DISP=OLD,DSN=ROGERS.JCL.VERS5(GENER)
000600 //TAPE DD DSN=(ROGERS.A),DISP=(NEW,KEEP),UNIT=3490
000700 //SYSIN DD DUMMY
000800 /*
CC0900 //
***** ***** Bottom of Data *****

```

Figure 12-12 Using the CREATE command to create a file in the file system

## The MOVE command

The **MOVE** command can now be used to move data from a z/OS UNIX file into the data being edited. To do this, specify the path name for the z/OS UNIX regular file as a parameter with the **MOVE** command. You can also specify the path name on the Edit/View Move panel that is displayed when no data source is specified with the **MOVE** command. If the path name for a directory is specified, the z/OS UNIX Directory List utility is invoked to allow you to select the regular file to be moved.

**Note:** The z/OS UNIX regular file is deleted after the data is moved.

The new syntax of the **MOVE** command is shown in Figure 12-13.

```
>>--MOVE-----><
      | -member----- | |---AFTER-----label-|
      | -(member)----- | | -BEFORE- |
      | -dsname----- |
      | -pathname----- |
```

Figure 12-13 Syntax of the MOVE command

Where pathname is used to specify the path name of a regular z/OS UNIX file to be moved.

Use the **MOVE** command to move a file into the file that is currently being edited, as shown in Figure 12-14 where the file named data in directory /u/rogers is moved into /u/rogers/cbprnt following the first statement.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT /u/rogers/cbprnt Columns 00001 00072
Command ==> move /u/rogers/data Scroll ==> HALE
***** ***** Top of Data *****
A00001 //COMPRESS JOB (999,P0K), 'PAUL ROGERS', CLASS=A, MSGCLASS=T,
```

Figure 12-14 MOVE command to move an existing file into a file being edited

**Note:** The file being moved is deleted. A warning window appears prior to the move to indicate this, as shown in Figure 12-15 on page 172.

```
Move has been requested for z/OS UNIX file:
  /u/rogers/rich.txt
Moved files are deleted.

Instructions:

  Press ENTER key to confirm move request.
  (Moved files will be deleted.)

  Enter END or EXIT command to return to the
  edit session without moving data.
```

Figure 12-15 Warning window regarding the MOVE command

## The REPLACE command

The **REPLACE** command can now be used to replace the data in a z/OS UNIX regular file using the data being edited. If the z/OS UNIX file does not exist, it is created.

The new syntax of the **REPLACE** command is shown in Figure 12-16.

```
>>---REPLACE-----><
| -REPL---- | | -member----- | | -labela--labelb- |
| -REP---- | | -(member)----- |
| | -dsname(member)- |
| | -dsname----- |
| | -pathname----- |
```

Figure 12-16 Syntax of the REPLACE command

Where `pathname` is used to specify the path name of a regular z/OS UNIX file to be replaced.

Figure 12-17 shows the REPLACE command that replaces the data in the file named `gener` at `/u/rogers` with the existing data in the edit screen `/u/rogers/cbprnt`. Before the replace takes place, a window appears with the following text.

```
Replace has been requested for data set:
 /u/rogers/gener
Data will be over-written.

Instructions:

  Press ENTER key to confirm replace request.
  (Data set will be replaced.)

  Enter END or EXIT command to return to edit
  session without replacing data.
```

Press the Enter key to do the replace.

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT /u/rogers/cbprnt Columns 00001 00072
Command ==> replace /u/rogers/gener Scroll ==> HALF
***** ***** Top of Data *****
CC0001 //ROGERSC JOB (POK,999),MSGCLASS=T,NOTIFY=ROGERS
000002 //PRINT EXEC PGM=IEBGENER
000003 //SYSPRINT DD SYSOUT=T
000004 //SYSUT2 DD SYSOUT=J
000005 //SYSUT1 DD DISP=OLD,DSN=ROGERS.JCL.VERS5(GENER)
000006 //TAPE DD DSN=(ROGERS.A),DISP=(NEW,KEEP),UNIT=3490
000007 //SYSIN DD DUMMY
000008 /*
CC0009 //
***** ***** Bottom of Data *****
```

Figure 12-17 REPLACE command to replace an existing file with the current file

## 12.4 EDIT macro command support

You can use edit macros, which look like ordinary editor commands, to extend and customize the editor. You create an edit macro by placing a series of commands into a data set or member of a partitioned data set. Then you can run those commands as a single macro by

typing the defined name in the command line. Edit macros can be either CLISTs or REXX EXECs written in the CLIST or REXX command language, or program macros written in a programming language (such as FORTRAN, PL/I, or COBOL).

The following EDIT macro commands now support z/OS UNIX file path names as sources or targets for data:

- ▶ COPY
- ▶ CREATE
- ▶ MOVE
- ▶ REPLACE

The macros have the same syntax as the EDIT primary commands described in 12.3, “EDIT primary commands support” on page 169.

Furthermore, the following EDIT macro commands are enhanced to support z/OS UNIX file names:

- ▶ DATASET - Now returns a z/OS UNIX file path name when editing a z/OS UNIX file
- ▶ BLKSIZE - Now returns the value of 0 (zero) when editing a z/OS UNIX file

## 12.5 ISPF services support

ISPF services help you develop interactive ISPF applications. These services can make your job easier because they perform many tedious and repetitious operations. In z/OS V1R9, the EDIT, BROWSE, VIEW and FILEXFER ISPF services are enhanced to support z/OS UNIX files.

### The EDIT ISPF service

The EDIT service can now be called by ISPF applications to edit the data in z/OS UNIX files. The new FILE parameter is used to pass the name of an ISPF variable with a value set to the path name of the z/OS UNIX file to be edited. The new RECLLEN parameter is used to specify a numeric value for the record length to be used when editing a z/OS UNIX file. This parameter causes the records to be loaded into the editor as fixed length and saved back in the file as fixed length.

The new syntax of the EDIT ISPF service is shown in Figure 12-18.

```
ISPEXEC EDIT FILE(file-var) [PANEL(panel-name)]
                             [MACRO(macro-name)]
                             [PROFILE(profile-name)]
                             [FORMAT(format-name)]
                             [MIXED(YES|NO)]
                             [LOCK(YES|NO)]
                             [CONFIRM(YES|NO)]
                             [WS(YES|NO)]
                             [PRESERVE]
                             [PARM(parm-var)]
                             [RECLLEN(rec-len)]
```

Figure 12-18 Syntax of the EDIT ISPF service

Where file-var is the name of an ISPF variable containing the path name of a z/OS UNIX regular file or directory, and rec-len is the record length to be used when editing the file.

## The BROWSE ISPF service

The BROWSE service can now be called by ISPF application to display the data in z/OS UNIX files. The new FILE parameter is used to pass the name of an ISPF variable with a value set to the path name of the z/OS UNIX file to be browsed. The new RECLLEN parameter is used to specify a numeric value for the record length to be used when browsing the z/OS UNIX file. This parameter causes new line characters in the data to be ignored as record delimiters.

The new syntax of the BROWSE ISPF service is shown in Figure 12-19.

```
ISPEXEC BROWSE FILE(file-var)    [PANEL(panel-name)]
                                  [FORMAT(format-name)]
                                  [MIXED(YES|NO)]
                                  [RECLLEN(rec-len)]
```

Figure 12-19 Syntax of the BROWSE ISPF service

The z/OS V1R9 changes are where `file-var` is the name of an ISPF variable containing the path name of a z/OS UNIX regular file or directory, and `rec-len` is the record length to be used when browsing the file.

## The VIEW ISPF service

The VIEW service can now be called by ISPF applications to display and manipulate the data in z/OS UNIX files. The new FILE parameter is used to pass the name of an ISPF variable with a value set to the path name of the z/OS UNIX file to be viewed.

The new syntax of the VIEW ISPF service is shown in Figure 12-20.

```
ISPEXEC VIEW FILE(file-var)    [PANEL(panel-name)]
                                  [MACRO(macro-name)]
                                  [PROFILE(profile-name)]
                                  [FORMAT(format-name)]
                                  [MIXED(YES|NO)]
                                  [CONFIRM(YES|NO)]
                                  [WS(YES|NO)]
                                  [CHGWARN(YES|NO)]
                                  [PARM(parm-var)]
```

Figure 12-20 Syntax of the VIEW ISPF service

Where `file-var` is the name of an ISPF variable containing the path name of a z/OS UNIX regular file or directory to view.

## The FILEXFER ISPF service

The FILEXFER service can now be called by ISPF applications to upload data into or download data from z/OS UNIX files to the workstation. The syntax of the FILEXFER ISPF service is unchanged from previous version. However, the value of the ISPF variable specified with the HOST parameter can now be set to the path name of the z/OS UNIX regular file to be uploaded or downloaded.

**Note:** For additional information, refer to *z/OS ISPF Services Guide*, SC34-4819.

## 12.6 PDF installation-wide data set allocation exit

The PDF installation-wide data set allocation exit allows you to create, delete, allocate, and deallocate data sets instead of using those functions provided by PDF. Note that allocations done by ISPF, the TSO ALLOCATE command, or TSO commands are not handled by the exit.

In z/OS V1R9, the PDF data set allocation exit is changed to support the allocation of z/OS UNIX files to a DD. Two additional parameters are now passed to the exit:

- Path name pointer** A pointer to an area of storage containing the absolute path name of the z/OS UNIX file to be allocated.
- Path name length** A full word binary integer that is the length of the z/OS UNIX file path name.

When the exit receives control due to an allocation request, the SVC 99 parameter list that is passed to the exit contains the following additional parameters:

- ▶ **PATHNAME**, key 8017  
Path name of the z/OS UNIX file to allocate. Path name is of the form /dev/fdnnn, where nnn is the file descriptor number. The real path name can be obtained via the path name pointer and path name length parameters. Maximum length for PATHNAME is 1023 bytes.
- ▶ **PATHOPT**, key 8018  
A 4-byte field containing the file options for the z/OS UNIX file.
- ▶ **FILEORG**, key 801D  
A 1-byte field indicating the organization of the z/OS UNIX file.

**Note:** Existing data set allocation exits will continue to function without any changes to process these additional parameters.

## 12.7 Support for editing ASCII data

With the increasing use of Java and WebSphere products on z/OS, more and more data is stored in ASCII files on z/OS. One example is XML documents for WebSphere Application Server. Prior to z/OS V1R9, there were few facilities available in z/OS to display and change ASCII data, especially under ISPF. In general, it is necessary for ISPF users to download their ASCII files to a workstation that supports the ASCII character set, edit the files on the workstation, and upload them back to the z/OS system.

In z/OS V1R9, an ASCII editing facility is provided through the ISPF editor. The ASCII editing facility translates ASCII data in a file to EBCDIC before displaying it at the terminal and translates EBCDIC data to ASCII when receiving input from the terminal to write to the file. A new **SOURCE** primary command for the ISPF editor is provided in z/OS V1R9 to control the ASCII editing facility.

To activate the ASCII editing facility for a file, perform these steps:

1. Start editing the file as you would for a normal EBCDIC file.
2. Then, enter the following command: **SOURCE ASCII**

After **SOURCE ASCII** is issued, the ISPF editor treats the source data as ASCII data and converts it from ASCII to the Coded Character Set ID (CCSID) of the terminal for display purposes. The data in the file remains unchanged. When you input or modify data at the



terminal, the ISPF editor translates the data entered from the CCSID of the terminal to ASCII before storing the data in the file.

To change back to the normal editing mode, where the data is not translated from and to ASCII when displaying and receiving input from the terminal, issue the primary EDIT command: **RESET SOURCE**.

**Note:** The ASCII editing facility uses the z/OS Unicode Conversion Services to translate the data between ASCII (CCSID 850) and the CCSID supported by the terminal. It is required that z/OS Unicode Conversion Services be installed and the required translations specified to it, in order for the ASCII editing facility to be operable.

### Handling line feed characters

Many times ASCII files contain line feed characters. When such an ASCII file is uploaded from the workstation to a fixed length data set, the data may not be structured correctly according to the line feed characters. The **LF** primary command is a new ISPF editor primary command that restructures the data in the file based on the line feed characters.

Figure 12-21 shows an ASCII file opened in the ISPF editor. As noticed, the data in the file is unreadable, since it is in the ASCII character set.

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      PELEG.SRC(TEST8) - 01.10                      Columns 00001 00072
Command ==> _____ Scroll ==> HALF
***** Top of Data *****
000001     ÉÁÈ /oÑ ã%/ÆÈ ã?É --ÑÁÏ-Ñ>ã%ÍÁÁ+/_Á
000002 /oÑÁ%/ÆÈ --/%Ñ/ÉÁÈ
000003 /oÑÁ%/ÆÈ --Ñ_ø?ÉÈÈ
000004 /oÑÁ%/ÆÈ --/ÈÈÈÑÁ
000005     Ñ>Á%ÍÁÁ ø/ÈÇ >/_Á
***** Bottom of Data *****

```

Figure 12-21 ASCII file in the ISPF editor without the ASCII editing facility

After starting the editor, we issue the **SOURCE ASCII** and **LF** primary commands in order to activate the ASCII editing facility and restructure the ASCII file according to the line feed characters it contains. The result is shown in Figure 12-22.

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      PELEG.SRC(TEST8) - 01.10                      Columns 00001 00072
Command ==> _____ Scroll ==> HALF
***** Top of Data *****
000001 /* set api flags for __iew_includeName() */
000002 apiflags.__aliases = 1;
000003 apiflags.__imports = 1;
000004 apiflags.__attrib = 1;
000005 /* include path name */
***** Bottom of Data *****

```

Figure 12-22 ASCII file in ISPF editor with the ASCII editing facility started (source ASCII command)

### LF command

The **LF** primary command allows you to realign the data being edited by interpreting the ASCII line feed character X'0A'.

**Note:** After issuing the **LF** command, if the data is saved, it is saved in the realigned state. There is no command to reverse the alignment. The command should *not* be executed twice against the data, because the blanks following the line feed character will be interpreted as part of the data for the next line.

## 12.8 Mixed case in ISPF command tables

Prior to z/OS V1R9, the ISPF Command Table Utility (option 3.9 from the primary ISPF menu) converted all characters in the Action field to upper case. This prevented users from defining commands with actions that required lower case characters or case sensitivity, such as z/OS UNIX commands.

In z/OS V1R9, ISPF allows you to define commands in ISPF command tables with lower case characters using the ISPF Command Table Utility. A new option field is added to the Command Table Utility – Extended Command Entry panel (panel ISPUICMX) to allow mixed case data entered in the Action field to be saved in the command table as mixed case. An example of this is shown in Figure 12-23 on page 179. If this option is not selected, data entered in the Action field is converted to upper case before being saved in the command table.

The column headings on the panel shown in Figure 12-23 on page 179 are:

- |               |   |
|---------------|---|
| <b>Verb</b>   | The command verb, which is the name of the command you are defining in the command table. A command verb must be 2 to 8 characters long, inclusive, and must begin with an alphabetic character. The content of this column is assigned to the ZCTVERB system variable.   |
| <b>Trunc</b>  | The minimum number of characters that you must enter to find a match with the command verb. If this number is zero or equal to the length of the command verb, you must enter the complete command verb. For example, in Figure 119 the PREPARE command has a truncation value of 4. Therefore, for the TST application used as the example in the figure, only the first four letters, PREP, must be entered to call this command. The content of this column is assigned to the ZCTTRUNC system variable. |
| <b>Action</b> | The actual coding of the action to be carried out when you enter the command. The action length must not be greater than 240 characters. The content of this column is assigned to the ZCTACT system variable.  |

For the mixed case support with z/OS V1R9, follow these steps to enter data on the panel shown in Figure 12-23 on page 179.

To enter or edit the coding for the action:

1. Enter the E command table line command to display the Extended Command Entry panel (ISPUICMX).
2. Type the required coding in the Action lines.

Normally, any text you type in lower case is translated to upper case before it is saved. To define some of the parameters in lower case, select the option **Allow mixed-case in Action field** on the Extended Command Entry panel. The case of the text you type is not translated and is saved as you input it.

**Note:** When you select the Allow mixed-case in Action field option:

- ▶ The first word must be input in upper case.
- ▶ If you use &ZPARM to obtain parameters from the command line, the parameters may be translated to upper case (regardless of the setting of the Allow mixed-case in Action field option).

3. Optionally, type a brief description of the purpose of the command in the Description lines.
4. Press PF3 to return to the Command Table Editing panel.

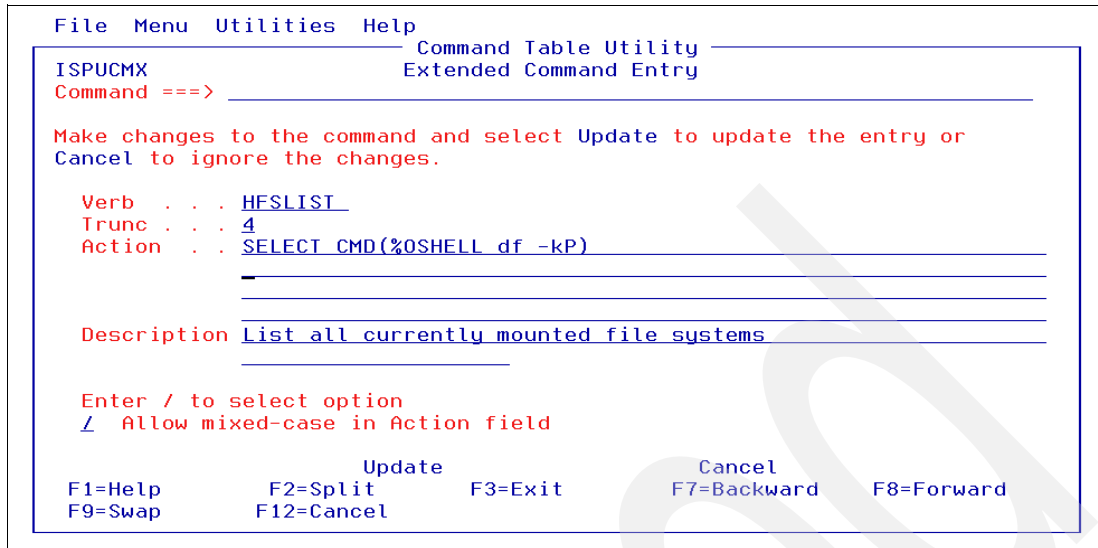


Figure 12-23 New field on the Command Table Utility - Extended Command Entry panel

The mixed case support is further extended to support Dialog Tag Language (DTL) CMD tags. A new MIXC keyword can now be specified with the CMDACT tag to prevent the command action specified with the CMD tag from being converted to upper case.

Archived



## Security enhancements

This chapter discusses the enhancements to security components in z/OS V1R9.

The following z/OS components are discussed:

- ▶ RACF
- ▶ Java Security API (JSec)
- ▶ z/OS Cryptographic Services - System SSL
- ▶ z/OS Cryptographic Services - PKI Services

## 13.1 RACF enhancements

### 13.1.1 Password phrase minimum length change

Password phrase support was introduced in z/OS V1R8. A *password phrase* is a character string consisting of mixed-case letters, numbers, and special characters including blanks. Password phrases have security advantages over passwords in that they are long enough to withstand most hacking attempts, and yet are unlikely to be written down because they are so easy to remember.

You can issue the **PHRASE** operand of the **ADDUSER** or **ALTUSER** command to assign a password phrase for a user. This enables the user to authenticate using a password phrase instead of a password when using an application that supports password phrases. For example:

```
ALTUSER ARUNDATI PHRASE('sm@llthings')
```

Currently, the only z/OS V1R9 application that supports user authentication using a password phrase is the Hardware Configuration Manager (HCM).

Password phrases can also provide an interoperability solution. Most platforms allow passwords longer than 8 characters. z/OS allows a maximum of 8 characters. Using password phrases, you can keep your z/OS password in sync with other platforms.

RACF enforces the following rules on the value of a new password phrase:

- ▶ Must be a text string of 14 to 100 characters
- ▶ Must not contain the user ID (as sequential upper case or sequential lower case characters)
- ▶ Must contain at least 2 alphabetic characters (A-Z, and a-z)
- ▶ Must contain at least 2 non-alphabetic characters (numerics, punctuation, or special characters)
- ▶ Must not contain more than 2 consecutive characters that are identical

These rules cannot be altered or overridden. The original decision on a minimum length of 14 characters for a password phrase is based on the premise that at least 14 characters are needed, with this limited set of rules applied to them, to be at least as secure as an 8 characters password with the **SETROPTS** password rules available.

However, if you want to use password phrase as an interoperability solution, you may run into problems, because you cannot set passwords with a length between 9 and 13 characters. This is due to the fact that normal passwords are limited to an 8-character maximum, and password phrases are limited to a 14-character minimum.

RACF in z/OS V1R9 allows you to change the minimum length of new password phrases to 9 characters. To do so, a new RACF exit, ICHPWX11, must be installed. With the ICHPWX11 RACF exit installed, the minimum length of password phrases is 9 characters. When the exit is not installed, the minimum stays 14 characters, as on z/OS V1R8.

The new password phrase exit is invoked by RACROUTE REQUEST=VERIFY processing and the **ADDUSER**, **ALTUSER**, **PASSWORD**, and **PHRASE** commands. The exit gains control when a new password phrase is processed, and can examine the value specified for the password phrase and enforce installation rules in addition to the RACF rules. For example, although RACF does not allow the user ID to be part of the password phrase, the exit could perform

more complex tests to also disallow the company name, the names of months, and the current year in the password phrase.

To allow you to easily perform your own tests on the password phrase, a sample ICHPWX11 exit is shipped in member RACEXITS in SYS1.SAMPLIB. The sample exit uses the new System REXX component in z/OS V1R9 to invoke a sample REXX exec. The sample REXX exec is shipped as member IRRPHREX in SYS1.SAMPLIB. The exit passes all its input information to the REXX exec, so you can perform all your installation-specific tests on the password phrase from within the REXX exec.

The REXX language is known for its strong string manipulation functions and therefore allows you to examine the password phrase easily. Furthermore, after the exit is installed, the REXX exec can be changed without requiring a re-IPL, as opposed to the other RACF exits.

To install the sample exit provided by IBM, do the following:

1. Install the ICHPWX11 exit in the link pack area so that RACF finds it during initialization.
2. Copy the REXX exec from member IRRPHREX in SYS1.SAMPLIB to SYS1.SAXREXEC.
3. Re-IPL your system.

If you change the password phrase quality rules that are coded in the IRRPHREX exec, you do not need to re-IPL. The changes you make to IRRPHREX take effect immediately when you save them. If you make changes to ICHPWX11, you must re-IPL to activate your changes.

### 13.1.2 Writable key ring functions

You can use RACF to create, register, store, and administer digital certificates and their associated private keys, and build certificate requests that can be sent to a certificate authority for signing. You can also use RACF to manage key rings of stored digital certificates. Digital certificates and key rings are managed in RACF primarily by using the **RACDCERT** command, or by using an application that invokes the **R\_data1ib** callable service (IRRSDL00 or IRRSDL64).

The **R\_data1ib** callable service provides an application programming interface to the Common Data Security Architecture (CSDA) data library functions, and is used by System SSL to establish secure sessions between servers. RACF has three categories for managing digital certificates:

▶ **User certificate**

This is a certificate that is associated with a RACF user ID and is used to authenticate the user's identity. The RACF user ID can represent a traditional user, or be assigned to a server or started procedure.

▶ **Certificate-authority certificate**

This is a certificate that is associated with a certificate authority and is used to verify signatures in other certificates.

▶ **Site certificate**

This is a certificate that is associated with an off-platform server or other network entity, such as a peer VPN server. This category of certificate can also be used to share a single certificate and its private key among multiple RACF user IDs. When used for sharing, a certificate may be referred to as a placeholder certificate.

Prior to z/OS V1R9, there was no mechanism available to applications that wanted to programmatically populate certificates in RACF.

In z/OS V1R9, the **R\_data1ib** callable service is enhanced with new functions to create or delete key rings, add or delete certificates to or from RACF, and connect or remove certificates from key rings. Using the new functions of the **R\_data1ib** callable service, applications can programmatically manage certificates in RACF.

Five new function codes are defined for the new **R\_data1ib** functions:

<b>X'07' NewRing</b>	Create a new key ring or remove all the certificates from an existing key ring.
<b>X'08' DataPut</b>	Add a certificate to the RACF database (if it does not already exist), and connect it to a key ring.
<b>X'09' DataRemove</b>	Remove a certificate from the key ring, and optionally delete it from the RACF database if the certificate is not connected to any other rings.
<b>X'0A' DelRing</b>	Delete a key ring.
<b>X'0B' DataRefresh</b>	Refresh the in-storage certificates in the RACF database if the DIGTCERT class is RACLISTed. If the DIGTCERT class is not RACLISTed, no action is performed. DataRefresh might be required after calling DataPut or DataRemove.

### The RDATA LIB RACF class

In addition to the new function codes, a new RACF class is now provided to allow more granular access control over the users of the **R\_data1ib** functions. The new RACF class name is RDATA LIB. The format of the profiles in the RDATA LIB class is:

```
<ring_owner>.<ring_name>.<function>
```

The new **R\_data1ib** functions in z/OS V1R9 are considered update functions. Therefore, their profiles are of the form:

```
<ring_owner>.<ring_name>.UPD
```

The older functions, prior to z/OS V1R9, only list the RACF database and therefore their profiles are of the form:

```
<ring_owner>.<ring_name>.LST
```

For a virtual key ring, the profile format is:

```
<ring_owner>.IRR_VIRTUAL_KEYRING.LST
```

For a virtual key ring owner, the possible IDs are SITE, CERTAUTH, or an ordinary RACF ID.

If the new profiles are absent, **R\_data1ib** reverts to checking authorization using the old IRR.DIGICERT.\* profiles in the FACILITY class.

For example, suppose that an application running under user ID CERTSRVR needs to be able to install certificates in all the key rings that start with MYC, and are owned by MYCJOB in the RACF database. Issue the commands shown in Figure 13-1 to authorize CERTSRVR:

```
RDEFINE RDATA LIB MYCJOB.MYC*.UPD UACC(NONE)
PERMIT MYCJOB.MYC*.UPD CLASS(RDATA LIB) ID(CERTSRVR) ACCESS(UPDATE)
```

Figure 13-1 RDATA LIB class RACF commands example



### 13.1.3 UTF8 characters support in digital certificates

As described in 13.1.2, “Writable key ring functions” on page 183, you can use RACF to store digital certificates. Prior to z/OS V1R9, RACF had a limitation that prevented certificates containing multibyte UTF8 characters in the Subject Distinguished Name or Subject Alternate Names of a certificate from being stored in the RACF database. This limitation prevented some applications from being able to store their certificates in the RACF database.

In z/OS V1R9, RACF allows you to store certificates containing multibyte UTF8 characters in the Subject Distinguished Name or Subject Alternate Names and for such certificates to be used for authentication to RACF.

However, a limitation still exists. To be able to store a certificate with multibyte UTF8 characters in the RACF database, the UTF8 characters must be convertible to characters in the IBM-1047 code page. The reason for this limitation is that RACF uses z/OS Unicode Conversion Services internally to convert the UTF8 characters in the Distinguished Names and Subject Alternate Names to IBM-1074 characters before storing them in the RACF database.

Certificates containing UTF8 characters are supported by the RACDCERT RACF command, the `initACEE` callable service and the `R_data lib` callable service.

**Note:** UTF8 support is not available for PKI Services yet. You cannot use a certificate with UTF8 characters as the PKI Services CA certificate.

### 13.1.4 REFRESH warning message after RACDCERT commands

On releases before z/OS V1R9, RACF did not issue a warning message indicating a `SETROPTS REFRESH` command is required to pick up changes in the DIGTCERT or DIGTNMAP classes even if they were RACLISTed. The lack of a warning message may cause users to forget to issue a `SETROPTS REFRESH` command after changing profiles in these RACF classes.

Therefore, in z/OS V1R9, RACF issues a warning message to remind you to issue a `SETROPTS REFRESH` command against the DIGTCERT and DIGTNMAP classes after a `RACDCERT` command. For example, assume that the DIGTCERT class is RACLISTed. Then issue the following command:

```
RACDCERT ADD(mycert) WITHLABEL('My Cert')
```

This message is issued by RACF:

```
IRR175I The new profile for DIGTCERT will not be in effect until a SETROPTS  
REFRESH has been issued.
```

The warning message is issued, if necessary, after RACDCERT GENCERT, ADD, MAP, DELETE, DELMAP, ALTER, REKEY and IMPORT.

**Note:** The DIGTRING class is not involved. Changes in key rings do not require `SETROPTS REFRESH` because in-storage key ring information is not used in any key ring-related process.

## 13.2 Java security API

Starting with z/OS V1R9, a Java interface is provided for performing RACF functions. The Java security API (JSec) provides easy integration into a security environment that runs other Java code, on or off z/OS, and provides easy mapping to the RACF user and group administration commands (such as **ADDUSER**, **ALTUSER**, **CONNECT**, and so on) while providing RACF extensibility.

JSec is shipped with z/OS V1R9 in two jar files in the HFS:

- ▶ /usr/include/java\_classes/userregistry.jar
- ▶ /usr/include/java\_classes/RACFuserregistry.jar

No other steps are required to install JSec support in z/OS V1R9. To use JSec on another platform, download the JSec jar files to that platform and include them on your CLASSPATH. The following are required to run a Java application that uses JSec:

- ▶ The client system where the JSec program is running on must be running at least Java(TM) 2 Runtime Environment V5.
- ▶ The z/OS system with the RACF database you wish to administer must be running IBM Tivoli Directory Server (LDAP server) configured with an SDBM back-end.

To compile a Java program that uses the JSec interface, the JSec jar files must be included in the program's CLASSPATH environment variable. In z/OS UNIX, this can be achieved by using the command shown in Figure 13-2:

```
export CLASSPATH=$CLASSPATH:/usr/include/java_classes/userregistry.jar:usr/include/java_
classes/RACFuserregistry.jar
```

Figure 13-2 Adding JSec jar files to the CLASSPATH

JSec returns RACF user and group data as *Attributes*, a common Java class used to describe pairs of names and values. Attribute names follow the format *segmentName\_fieldName*. For example, the attribute name for the UID field in the OMVS segment is *OMVS\_UID*.

### JSec classes

The JSec Java classes are packaged in two packages:

- ▶ com.ibm.security.userregistry
- ▶ com.ibm.eserver.zos.racf.userregistry

Package com.ibm.security.userregistry contains three interfaces:

- ▶ SecAdmin

This is the interface for security administration of users and groups.

- ▶ User

This interface extends java.security.Principal by providing methods to get the attributes of user and the groups that the user belongs to.

- ▶ UserGroup

This interface extends java.security.acl.Group to return attributes of the group and to allow group membership to have qualifying attributes. UserGroup is intended to be a group of individual users.

Package `com.ibm.security.userregistry` also contains an Exception `SecAdminException`, which is a super class of all exceptions thrown from this package.

Package `com.ibm.eserver.zos.racf.userregistry` contains the following classes:

- ▶ `RACF_Group`  
This class implements `UserGroup` interface for RACF groups.
- ▶ `RACF_remote`  
`RACF_remote` defines characteristics and connection parameters of a remote RACF to be accessed via LDAP/SDBM.
- ▶ `RACF_SecAdmin`  
This class implements `SecAdmin` interface to RACF and provides additional utility methods (including cloning a user ID, and the ability to display attributes in alphabetical order).
- ▶ `RACF_User`  
This class implements the `User` interface for RACF users.
- ▶ `RACFAttribute`  
This is a data structure that keeps track of the data, behavior, and so on that is associated with each RACF attribute.
- ▶ `Segment`  
This is a simple data class that keeps three pieces of information for each RACF non-base segment.

The complete Javadoc™ for the JSec API classes and sample programs can be found at:

<http://www-03.ibm.com/servers/eserver/zseries/software/java/>

### Example

Figure 13-3 on page 188 shows an example of how to use the JSec interface to connect to a remote RACF database and list the attributes for user `JAVA1`. We use the `RACF_remote` class constructor to connect to a z/OS LDAP server with an SDBM back-end running on our test system `SC60`. System `SC60`'s DNS address is `wtsc60.itso.ibm.com`, and the z/OS LDAP server SDBM back-end is listening on port 3389. User `JAVA1` with password `JAVA1` is used for the LDAP bind. The SDBM suffix is `O=ITSORACF`.

After a connection with RACF on `SC60` is established, we use the `RACF_SecAdmin` class to define a RACF administrator. The RACF administrator is then used to create a `User` object for user `JAVA1`. The `User` object is used to get `JAVA1`'s attributes from the RACF database. Finally, `JAVA1`'s RACF attributes are output using the `displayAttributes()` method of the `RACF_SecAdmin` object.

```

import com.ibm.eserver.zos.racf.userregistry.*;
import com.ibm.security.userregistry.*;
import javax.naming.*;
import javax.naming.directory.*;

public class TestJSec {

    public static void main(String[] args) {

        RACF_remote SC60_RACFdb = new RACF_remote(
            "ldap://wtsc60.itso.ibm.com:3389",
            "simple",
            "JAVA1",
            "JAVA1", // password during testing
            "o=itsoracf"
        );

        try {
            SecAdmin racfAdmin = new RACF_SecAdmin(SC60_RACFdb);
            if (racfAdmin != null) {
                User JAVA1 = racfAdmin.getUser("JAVA1");

                BasicAttributes attrJAVA1 = JAVA1.getAttributes();

                System.out.println("Attributes returned for JAVA1:");
                RACF_SecAdmin.displayAttributes(attrJAVA1);
            }
        }
        catch (Exception e) {
            System.out.println("Exception in TestJSec.java " +
                e.getMessage() + "\n");
            e.printStackTrace();
        }
    }
}

```

*Figure 13-3 Using JSec to list user attributes*

The output from this example is shown in Figure 13-4 on page 189. Note that some attributes are shown with the value `No values`. These are attributes with a Boolean value, meaning they either exist for the RACF entity or do not.

For example, a `BASE_OPERATIONS` attribute exists for user `JAVA1`. This attribute has no value; it is only there to indicate that the user has the `OPERATIONS` attribute in RACF.

```
PELEG @ SC65:/u/peleg>java TestJSec
Attributes returned for JAVA1:
BASE_CREATED: 03.268
BASE_DAYS: ANYDAY
BASE_DFLTGRP: SYS1
BASE_LAST-ACCESS: 07.149/15:17:38
BASE_NAME: JAVA1 USER
BASE_OPERATIONS: No values
BASE_OWNER: WELLIE2
BASE_PASS-INTERVAL: 254
BASE_PASSDATE: 07.149
BASE_PASSWORD: Password Exists
BASE_SPECIAL: No values
BASE_TIME: ANYTIME
BASE_USERID: JAVA1
OMVS: No values
OMVS_HOME: /u/java1
OMVS_PROGRAM: /bin/sh
OMVS_UID: 000000000
TSO: No values
TSO_ACCTNUM: MVS
TSO_COMMAND: ISPPDF
TSO_MAXSIZE: 00000000
TSO_PROC: BPXPROC
TSO_SIZE: 02000000
TSO_UNIT: SYSDA
```

Figure 13-4 JSec program output

## 13.3 System SSL enhancements

System SSL is part of the System SSL Cryptographic Services base element of z/OS. Some parts of System SSL ship in HFS files and some in PDS and PDSE data sets. System SSL has three FMIDs:

- ▶ HCPT390 - Cryptographic Services System SSL (base element)
- ▶ JCPT391 - Cryptographic Services Security Level 3 (optional priced feature)
- ▶ JCPT39J - Japanese

### 13.3.1 Introduction to the SSL protocol

Secure Sockets Layer (SSL) is a communications protocol that provides secure communications over an open communications network, for example, the Internet. The SSL protocol is a layered protocol that is intended to be used on top of a reliable transport, such as TCP/IP.

SSL provides data privacy and integrity, as well as server and client authentication based on public key certificates. After an SSL connection is established between a client and server, the SSL protocol transparently adds encryption and integrity to the data communications between the client and the server. System SSL supports the SSL V2.0, SSL V3.0 and Transport Layer Security (TLS) V1.0 protocols. TLS V1.0 is the latest version of the secure sockets layer protocol.

System SSL provides a set of SSL C/C++ callable application programming interfaces that, when used with the z/OS Sockets APIs, provide the functions required for applications to establish this secure sockets communication.

In addition to providing the API interfaces to exploit the SSL and TLS protocols, System SSL is also providing a suite of Certificate Management APIs. These APIs give the capability to create and manage your own certificate databases, utilize certificates stored in key databases, key rings or tokens for purposes other than SSL, and to build and process PKCS #7 standard messages.

In addition to providing APIs for applications to use for both SSL and certificate management support, System SSL also provides a certificate management utility called `gskkyman`. This utility allows for the management of certificates stores in a key database file or z/OS PKCS #11 token.

### **An overview of how SSL works**

The SSL protocol begins with a *handshake*. During the handshake, the client authenticates the server, the server optionally authenticates the client, and the client and server agree on how to encrypt and decrypt information. In addition to the handshake, SSL also defines the format used to transmit encrypted data.

X.509 (V1, V2 or V3) certificates are used by both the client and server when securing communications using System SSL. The client must verify the server's certificate based on the certificate of the Certificate Authority (CA) that signed the certificate, or based on a self-signed certificate from the server. The server must verify the client's certificate (if requested) using the certificate of the CA that signed the client's certificate. The client and the server then use the negotiated session keys and begin encrypted communications.

The SSL protocol runs above the TCP/IP and below higher-level protocols such as HTTP. It uses TCP/IP on behalf of the higher-level protocols. Figure 13-5 on page 191 shows a schematic description of the SSL protocol.

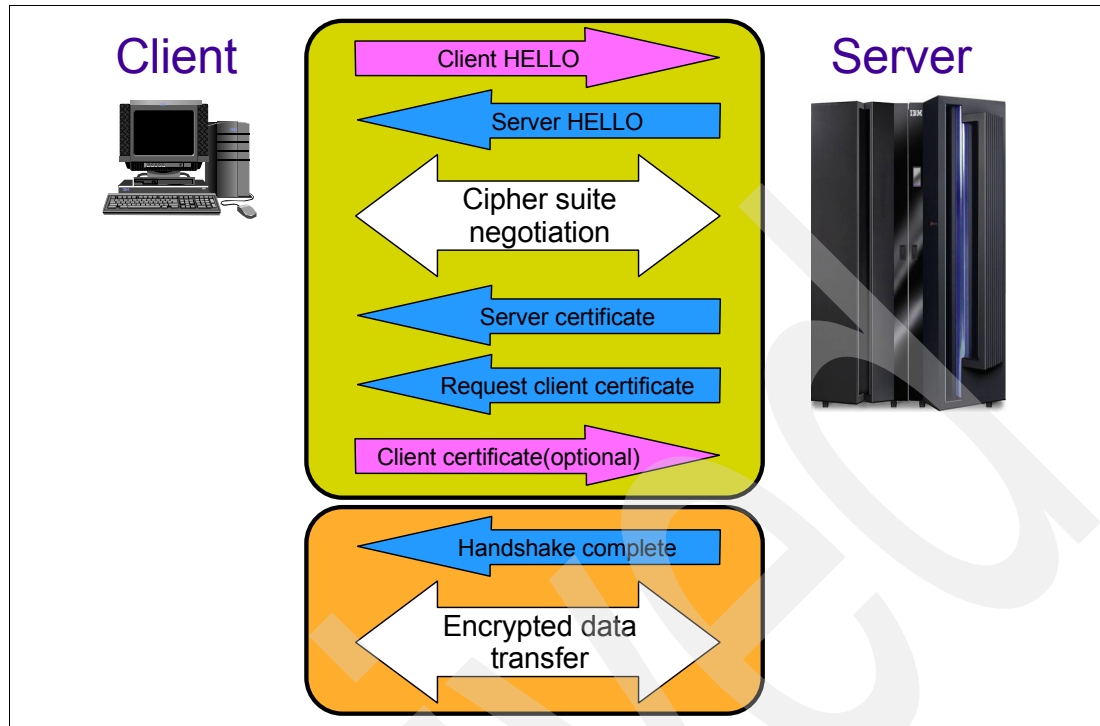


Figure 13-5 Overview of the SSL protocol

The capabilities of SSL address several fundamental concerns about communication over the Internet and other TCP/IP networks:

- ▶ **SSL server authentication**

This allows a client application to confirm the identity of the server application. The client application through SSL uses standard public-key cryptography to verify that the server's certificate and public key are valid and has been signed by a trusted CA that is known to the client application.

- ▶ **SSL client authentication**

This allows a server application to confirm the identity of the client application. The server application through SSL uses standard public-key cryptography to verify that the client's certificate and public key are valid and have been signed by a trusted CA that is known to the server application.

- ▶ **An encrypted SSL connection**

This requires all information being sent between the client and server application to be encrypted. The sending application is responsible for encrypting the data and the receiving application is responsible for decrypting the data. In addition to encrypting the data, SSL provides message integrity. Message integrity provides a means to determine whether the data has been tampered with since it was sent by the partner application.

### 13.3.2 Certificate revocation lists (CRLs) granularity

System SSL supports X.509 certificates and certificate revocation lists (CRLs). CRLs provide a means for certificates to be revoked prior to their expiration time. Certificate revocation may be required due to several reasons; for example, the issuer of the certificate is no longer trustable, or the keys associated with the certificate have been compromised. System SSL supports CRLs that are stored in LDAP directories.

Prior to z/OS V1R9, when System SSL was performing validation of a certificate against CRL entries in an LDAP directory, System SSL did not require a CRL entry to exist for a certificate. Furthermore, System SSL would fail the validation of a certificate if CRL checking was specified and the application could not contact (bind to) the LDAP server. The application had no control of how to enforce the use of CRLs during the validation process. However, some applications may require a more strict enforcement, while other applications may be satisfied with a less restricted enforcement.

To give more flexibility to the applications, a new tuning environment variable and environment attribute is added in z/OS V1R9 to define the SSL environment. The tuning variables allow the application to specify one of the following:

- |                        |   |
|------------------------|---|
| <b>High security</b>   | Certificate validation requires the LDAP server to be contactable and a CRL to be defined.  |
| <b>Medium security</b> | Certificate validation requires the LDAP server to be contactable, but does not require a CRL to be defined. This is the same behavior as previous releases and is the default. |
| <b>Low Security</b>    | Certificate validation will not fail if the LDAP server cannot be contacted.  |

The new CRL controls are set at the SSL environment level and apply to all secure connections within that SSL environment. An SSL environment defines common attributes that will apply to all the application's secure connections.

There are two ways to set the CRL security granularity level:

- ▶ Setting the environment variable `GSK_CRL_SECURITY_LEVEL` to `HIGH`, `MEDIUM` or `LOW`. Figure 13-6 shows an example of how to set the environment variable to `HIGH`:

```
export GSK_CRL_SECURITY_LEVEL=HIGH
```

Figure 13-6 Setting the CRL security level using an environment variable

Refer to Appendix A of *System SSL Programming Guide*, SC24-5901, for more information about environment variable settings.

- ▶ Using the SSL API function `gsk_attribute_set_enum` to set attribute `GSK_CRL_SECURITY_LEVEL` to one of the following:
  - `GSK_CRL_SECURITY_LEVEL_HIGH`
  - `GSK_CRL_SECURITY_LEVEL_MEDIUM`
  - `GSK_CRL_SECURITY_LEVEL_LOW`

Figure 13-7 shows an example of setting the `GSK_CRL_SECURITY_LEVEL` attribute to `HIGH`.

```
rc = gsk_attribute_set_enum(  
    env_handle,  
    GSK_CRL_SECURITY_LEVEL,  
    GSK_CRL_SECURITY_LEVEL_HIGH  
);
```

Figure 13-7 Setting the CRL security level using the SSL API



For more information about parameters of the `gsk_attribute_set_enum` API function, refer to *System SSL Programming Guide*, SC24-5901, as well as the SSL header file `/usr/lpp/gskssl/include/gskssl.h`.

### 13.3.3 Rehandshake notification

Prior to z/OS V1R9, after an SSL handshake had completed establishing a secure connection, the application could cause a rehandshake to occur in order to have new session keys established for the connection or to have the session's cipher algorithm reset. This rehandshake is known as a *renegotiation of the secure connection*. Either side of the secure connection can initiate a rehandshake. The TLS Protocol Version 1.0 RFC (RFC 2246) allows application data to flow during the handshake renegotiation. However, some implementations do not tolerate data flow during the renegotiation.

In z/OS V1R9, rehandshake notification callback support allows applications to suspend the flow of data until the rehandshake is complete, if they desire to do so, and thus be able to support more implementations of the SSL/TLS protocol.

In order for an application to be notified when a handshake is being renegotiated or completed, the application is required to register callback routines with System SSL. Callback routines are routines that reside in the application and are passed control by System SSL when a particular condition is met or a certain event occurs. In this case, the callback functions receive control when a secure connection is being renegotiated or when the renegotiation is completed.

Two rehandshake notification functions are provided by System SSL. The first callback function is intended to give applications the capability of knowing when a rehandshake is occurring. The second function gives applications the capability of knowing when a rehandshake has completed. Effectively, these two callback functions also allow applications to know when the connection is being renegotiated, and when it is again ready for normal secure communications.

**Note:** This support is limited to the SSL V3.0 and TLS V1.0 protocols. SSL V2.0 does not support renegotiation of established secure connections.

The identification of the callback routines is done through the `gsk_attribute_set_callback` API function. Use the `gsk_reset_callback` structure to point to the application's callback functions. Then, use `gsk_attribute_set_callback` to identify the functions to System SSL.

An example is shown in Figure 13-8 on page 194. This example focuses on statistics gathering, but the same concept can be used to control the flow of application data during a handshake renegotiation. The callback routines can suspend application data flow when the rehandshake is initiated, and resume data flow after the rehandshake complete callback is called.

```

/* Callback 1 - Rehandshake init */
int rehandshakes_initiated = 0;
void Reset_Init(gsk_handle con_handle) {
    rehandshakes_initiated++;
};

/* Callback 2 - Rehandshake complete */
int rehandshakes_completed = 0;
void Reset_Complete(gsk_handle con_handle) {
    rehandshakes_completed++;
};

...

gsk_reset_callback rehandshake_callbacks;

rehandshake_callbacks = {Reset_Init, Reset_Complete};

rc = gsk_attribute_set_callback(
    env_handle,
    GSK_SESSION_RESET_CALLBACK,
    rehandshake_callbacks
);

```

Figure 13-8 Setting System SSL rehandshake callback functions

In the example, whenever a rehandshake is initiated, the application's `Reset_Init` function is called. When the handshake has completed, the `Reset_Complete` function is called.

### 13.3.4 Host name validation

System SSL provides an API function, `gsk_validate_server`, for applications to validate a server's certificate by specifying the host name associated with the server. The server certificate must contain the specified server's host name as either the common name (CN) element of the subject name or as a DNS entry for the subject alternate name extension. System SSL first compares the host name against the value of the CN in the certificate and if no match is found, proceeds to compare against the DNS value in the subject alternate name extension.

The RFC for HTTP over TLS (RFC 2818) and several other Internet protocol standards require that the DNS subject name in the subject alternate name extension be compared to first and only if that is not present, a check against the CN in the certificate should be performed. Prior to z/OS V1R9, this was not achievable by the System SSL API.

In z/OS V1R9, System SSL adds a new API function that gives applications the flexibility to specify the order and how comparisons are to be performed against the host name and the X.509 certificate. The new API function is called `gsk_validate_hostname`. It allows you to specify a value option that controls the composition and the order of the host name validation process. Four options are available:

- ▶ `GSKCMS_VALIDATE_HOSTNAME_CN`  
This validates the host name against the CN of the certificate first and then against the DNS entry for the subject alternate name extension if no match is found in the CN.
- ▶ `GSKCMS_VALIDATE_HOSTNAME_CN_ONLY`  
This validates the host name against the CN of the certificate only.

- ▶ **GSKCMS\_VALIDATE\_HOSTNAME\_DNS**  
This validates the host name against the DNS entry in the subject alternate name extension first and, only if that is not present, validates the host name against the CN.
- ▶ **GSKCMS\_VALIDATE\_HOSTNAME\_DNS\_ONLY**  
This validates the host name against the DNS entry in the subject alternate name extension only.

Figure 13-9 shows an example of using `gsk_validate_hostname` to set the validation process to check the DNS entry first and then CN.

```
#include <gskcms.h>

gsk_validate_hostname(
    x509_certificate,
    hostname,
    GSKCMS_VALIDATE_HOSTNAME_DNS
);
```

Figure 13-9 Controlling the System SSL host name validation process

### 13.3.5 Hardware-to-software switch notification

System SSL uses the Integrated Cryptographic Service Facility (ICSF) if it is available. ICSF provides hardware cryptographic support which will be used instead of the System SSL software algorithms. System SSL checks for the hardware support during its runtime initialization processing, and will use the hardware support if available.

System SSL also takes advantage of the CP Assist for Cryptographic Function (CPACF) when available. CPACF is a set of cryptographic instructions available on all CPs of a z990, z890, and z9 EC and z9 BC processors. The SHA-1 algorithm is always available. The SHA-256 algorithm is available on the z9 EC and z9 BC. CPACF DES/TDES Enablement, feature 3863, provides for clear key DES and TDES instructions. On the z9 EC and z9 BC, this feature also includes clear key AES for 128-bit keys. ICSF is not required in order for System SSL to use the CPACF.

On top of that, System SSL contains software implementations for every cryptographic algorithm that it requires. If a severe ICSF error occurs during a cryptographic operation, System SSL stops using the hardware support and reverts to using the software algorithms. The switch to software support is done transparently to the application. Prior to z/OS V1R9, System SSL did not issue a notification of the switch to software support, so no corrective action could have been taken by the system programmers.

Starting with z/OS V1R9, the System SSL started task, GSKSRVR, issues two new notification messages when a switch occurs. The messages are:

```
GSK01051E jobname/ASID - Hardware encryption error. ICSF hardware encryption
processing is unavailable
```

```
GSK01052W jobname/ASID - Hardware encryption error. algorithm encryption
processing switched to software
```

In order to provide notification about the switch from hardware to software, the System SSL started task GSKSRVR must be started before any application using System SSL. This allows System SSL to detect the switch and inform the started task of the event. The GSKSRVR started task initially displays message GSK01051E to inform about the switch from hardware to software. The GSKSRVR system log contains more detailed messages

about the actual encryption function that converted to System SSL's software implementation.

These messages provide the system programmers with an indication that the switch has occurred. The system programmers need to ensure that ICSF hardware encryption services are up and functioning correctly.

**Note:** To utilize hardware encryption again, after the problem is fixed, a restart of the SSL application or process is needed. The GSKSRVR started task does not need to be restarted.

## 13.4 PKI Services enhancements

z/OS Cryptographic Services PKI Services allow you use z/OS to establish a PKI infrastructure 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. The users in the organization can use a PKI Services application to request and obtain certificates through their own Web browsers, while the authorized PKI administrators approve, modify, or reject these requests through their own Web browsers.

The Web applications provided with PKI Services are highly customizable, and a programming exit is also included for advanced customization. You can allow automatic approval for certificate requests from certain users and, to provide additional authentication, add host IDs, such as RACF user IDs, to certificates you issue for certain users. You can also issue your own certificates for browsers, servers, and other purposes, such as virtual private network (VPN) devices, smart cards, and secure e-mail.

PKI Services supports Public Key Infrastructure for X.509 version 3 (PKIX) and Common Data Security Architecture (CDSA) cryptographic standards. It also supports the following:

- ▶ The delivery of certificates through the Secure Sockets Layer (SSL) for use with applications that are accessed from a Web browser or Web server.
- ▶ The delivery of certificates that support the Internet Protocol Security standard (IPSEC) for use with secure VPN applications or IPSEC-enabled devices.
- ▶ The delivery of certificates that support Secure Multipurpose Internet Mail Extensions (S/MIME), for use with secure e-mail applications.

z/OS is certified as IdenTrust-compliant. This allows z/OS installations to participate in the IdenTrust infrastructure by configuring PKI Services to operate as an IdenTrust compliant certificate authority (CA).

### 13.4.1 Automatic certificate renewal processing

Prior to z/OS V1R9, certificate owners had to manually renew their certificates after they received an expiration notification. To make renewal of certificates easier, PKI Services in z/OS V1R9 support automatically sending a renewed certificate to the owner before the old certificate expires.

Automatic certificate renewal is done through the PKI Services configuration file, `pkiserv.conf`, and the template file, `pkiserv.tmpl`.

The relevant parameters in `pkiserv.conf` are:

► `ExpireWarningTime=`

This parameter indicates how soon before certificate expiration to send a warning message or a renewed certificate (that is, the number of days or weeks before the day and time the certificate expires). If automatic certificate renewal is active, this parameter indicates how soon before certificate expiration to renew the certificate and send it to the owner.

This parameter is optional. Its absence indicates no expiration checking is performed and no automatic certificate renewal occurs. Also, if the parameter is present but has an incorrect value or if PKI Services is configured to operate without LDAP, no expiration checking or automatic certificate renewal is done.

► `RenewCertForm=`

The full path name or data set name containing the “renewed certificate”. Defaults to no certificate sent.

In addition, update the `pkiserv.tmpl` file with the following:

► The `AUTORENEW` tag

This tag determines whether the certificate is to be automatically renewed when it approaches expiration. This tag has the form `<AUTORENEW=value>`, where value can have the value Y, y, N, or n. If the `AUTORENEW` tag has any other value, or does not immediately follow the `NICKNAME` tag, PKI Services operates as if the tag is not present. The tag has the following meanings:

- `AUTORENEW` tag not present means that the certificate is not set up for automatic renewal.
- `AUTORENEW=Y` means that the certificate is enabled for automatic renewal. `AUTORENEW=N` means that the certificate is eligible for automatic renewal, but automatic renewal is disabled.

**Note:** Adding the `AutoRenew=Y` tag does not enable all certificates to be automatically renewed. It enables only the newly issued certificates after the template is been updated.

► The notification e-mail address of the receiver

A renewed certificate gets all the information from the original certificate, but with a new expiration date which is set from the expiration date of the original certificate plus the `NotAfter` value from template. Furthermore, exit hooks can be set up to add criteria to disallow automatic renewal and to perform post processing for the renewed certificate.

### Example

We want to set up PKI Services so that all the certificates generated from the “1-Year PKI SSL Browser Certificate” template are renewed 30 days before they expired. We update the `pkiserv.conf` file, as shown in Figure 13-10.

```
ExpireWarningTime=30d
RenewCertForm=/etc/pkiserv/renewcertmsg.form
```

Figure 13-10 Example updates to `pkiserv.conf`

We also update the `pkiserv.tmpl` file, as shown in Figure 13-11 on page 198.

```

<TEMPLATE NAME=1-Year PKI SSL Browser Certificate>
<TEMPLATE NAME=PKI Browser Certificate>
<NICKNAME=1YBSSL>
  <AUTORENEW=Y>
  ...
  %%NotifyEmail%%

```

Figure 13-11 Example updates to `pkiserv.tpl`

Now 30 days before the certificate expires, an e-mail with a renewed certificate will be sent.

### 13.4.2 RACF-style distinguished name

The `AuthName1` parameter in `pkiserv.conf` is used to specify the LDAP administrator's Distinguished Name (DN). The value specified in the `AuthName1` parameter is used by PKI Services to perform LDAP binding. LDAP binding is the process of authenticating a DN to the LDAP server. The DN is used to identify unambiguously a collection of attributes in the directory.

In z/OS V1R9, the LDAP administrator DN specified in the PKI Services configuration file can now be a RACF-style DN, underlying RACF data. A RACF-style DN for a user or group contains two required attributes plus a suffix. The required attributes are `racfid`, which specifies the user ID or group ID, and `profiletype`, which specifies user or group. The suffix specifies the SDBM suffix. The suffix for SDBM may contain additional attributes.

To use the new support, explicitly specify a RACF-style DN for the `AuthName1` parameter in `pkiserv.conf`. An example is shown in Figure 13-12.

```

AuthName1=RACFID=ADMIN,PROFILETYPE=USER,O=RACFDB,C=US
AuthPwd1=secret

```

Figure 13-12 Specifying a RACF-style DN in `pkiserv.conf`

You can also implicitly specify a RACF-style DN for the PKI Services LDAP administrator using the `BindProfile1` parameter in `pkiserv.conf`. For example, to define an LDAP bind profile named `MY.LDAP.SERVER1` in RACF with the equivalent attributes as the previous example, use the RACF command in Figure 13-13, then specify `BindProfile1=MY.LDAP.SERVER1` in `pkiserv.conf`.

```

RDEFINE LDAPBIND MY.LDAP.SERVER1 PROXY(LDAPHOST(ldap://some.ldap.host:389)
BINDDN('RACFID=ADMIN,PROFILETYPE=USER,O=RACFDB,C=US') BINDPW('secret'))

```

Figure 13-13 Specifying a RACF-style DN using a RACF bind profile

### 13.4.3 E-mail notification for administrators

In z/OS V1R9, PKI Services support sending an e-mail notification to the PKI Services administrators to notify them of requests waiting for their approval.

Otherwise, the PKI Services administrators have no way of knowing about new requests, other than by logging on to the PKI Services Web application periodically.

New parameters are added to pkiserv.conf for the new e-mail notification support:

- ▶ AdminNotifyNew<n>  
The e-mail address to which notification should be sent immediately when a request is created and requires approval. The notification is only sent once. There can be multiple entries, where <n> is 1 for the first entry and increases sequentially for additional entries.
- ▶ AdminNotifyReminder<n>  
The e-mail address to which reminder notifications of requests pending approval should be sent when PKI Services starts, and once a day thereafter. There can be multiple entries, where <n> is 1 for the first entry and increases sequentially for additional entries.
- ▶ AdminNotifyForm  
The full path name or data set name containing the request(s) pending for approval message form. Defaults to no notification sent.

It is possible to use AdminNotifyNew, AdminNotifyReminder, or both.

#### 13.4.4 Longer validity period for certificates

Starting with z/OS V1R9, PKI Services allows specifying a certificate validity period greater than 10 years, which was the limitation on previous releases. The new limit is 9999 days, which is about 27 years.

This enhancement allows you to define certificates with longer validity periods. For example, to set up a template for certificates with validity for 20 years, specify the following in pkiserv.tmpl, as shown in Figure 13-14.

```
<CONSTANT>  
%%NotAfter=7300%%  
</CONSTANT>
```

Figure 13-14 Example of specifying a certificate validity period

The **R\_PKIServ** callable service supports a NotAfter parameter for the GENCERT, REQCERT, GENRENEW and REQRENEW function codes. The NotAfter parameter now also supports values greater than 10 years.

#### 13.4.5 Query on expiring certificates

The PKI Services administration Web pages allow PKI administrators to process certificate requests, preregistration records, and individual certificates. In addition, the PKI Services administration Web pages allow administrators to perform searches for certificate requests, preregistration records, and certificates.

Prior to z/OS V1R9, there was no interface allowing administrators to search for certificates based on the number of days they will expire in. So the administrators could not easily know which certificates were about to expire.

In z/OS V1R9, the administration Web pages are enhanced to allow administrators to search certificates using future days until expiration as a search criteria. Figure 13-15 on page 200 shows the new Web page for the administrator to perform searches on certificates and certificate requests.

• **Specify search criteria for certificates and certificate requests**

<p><b>Certificate Requests</b></p> <ul style="list-style-type: none"> <li><input type="radio"/> Show all requests</li> <li><input type="radio"/> Show requests pending approval</li> <li><input type="radio"/> Show approved requests</li> <li><input type="radio"/> Show completed requests</li> <li><input type="radio"/> Show rejected requests</li> <li><input type="radio"/> Show rejections in which the client has been notified</li> <li><input type="radio"/> Show preregistered requests</li> </ul>	<p><b>Issued Certificates</b></p> <ul style="list-style-type: none"> <li><input type="radio"/> Show all issued certificates</li> <li><input type="radio"/> Show revoked certificates</li> <li><input type="radio"/> Show suspended certificates</li> <li><input type="radio"/> Show expired certificates</li> <li><input checked="" type="radio"/> Show active certificates (not expired, not revoked, not suspended)</li> <li><input type="radio"/> Show disabled certificates (suspended or revoked, not expired)</li> <li><input type="radio"/> Show automatic renewal enabled certificates</li> <li><input type="radio"/> Show automatic renewal capable certificates</li> </ul>
---	--

**Additional search criteria (Optional)**

Requestor's name

Show recent activity only

Show certificates that will expire  (Only applicable to active certificates when recent activity is not selected)

Figure 13-15 Web page for searching certificates that are about to expire





## z/OS Communication Server

This chapter describes some of the z/OS Communication Server enhancements introduced in z/OS Version 1 Release 9:

- ▶ zIIP-assisted IPsec
- ▶ Policy-based routing

For complete information about all the new functions introduced in this release (as well as the past three releases) regarding IP and SNA functions, refer to *z/OS Communication Server: New Function Summary*, GC31-8771.

## 14.1 zIIP-assisted IPSec

zIIP-assisted IPSec enables z/OS Communications Server to direct CPU-intensive IPSec processing to an IBM System z9 Integrated Information Processor (zIIP). In z/OS V1R9 Communications Server, this function can lower the computing cost incurred by the IPSec protocols, while at the same time increasing the processing capacity of your general purpose central processors (CPs).

This function can also be enabled on machines with no zIIPs, so that you can project the effectiveness of zIIP for your current IPSec workload. When this function is enabled on a z/OS server with no zIIPs, MVS accounts for the zIIP-eligible workload that was processed on CPs, in SMF record types 30 and 7x. You can use this accounting information to project the percentage of your workload that would be zIIP-eligible, if you had zIIPs configured to your MVS image.

IBM System z9 Integrated Information Processor (zIIP) was introduced at the beginning of 2006. zIIP is a specialty engine designed to free up general purpose CPs and lower the software costs for selected workloads.

In the following text, the word “ipse” is spelled in two different ways with different meanings, as follows:

<b>IPSec</b>	This refers to the Virtual Private Network (VPN) IP Security (IPsec), a peer-to-peer IP tunnel.
<b>IPSEC</b>	This refers to the IPSEC feature in z/OS Communication Server that provides TCP/IP filtering (firewall) and VPN IPsec.

Communication Server's IPSec is the second IBM's exploiter of the zIIP assist processor. The first one was DB2 V8. It uses the System z9 crypto hardware, Crypto Express2 (CE2), for the following:

- ▶ Data encryption and decryption
- ▶ Authentication

Even using the CE2, some bulk workloads such as FTP or Tivoli Storage Manager (TSM) can utilize a significant amount of CPU. The cost of CPU is relative to the amount of data being moved.

In some cases, when running an LPAR at a high utilization level, the usage of IPSec could be a problem. In most of the cases the TCP/IP address space has a high priority and depending on the workload priority, this problem can be magnified. To minimize the problem the IPSec workload could run on a different TCP/IP address space with a lower priority, but the CPU consumption problem will remain.

For more information on the zIIP configuration, see and perform a search on zIIPs:

<http://www.ibm.com/support/techdocs>

### 14.1.1 Implementation of zIIP-assisted IPSEC

A new option was included in the GLOBALCONFIG statement to enable the SRB-mode IPSec Authentication header (AH) and Encapsulating Security Payload (ESP) protocols to be processed on zIIP. Figure 14-1 on page 203 shows the GLOBALCONFIG statement to configure the zIIP-assisted IPSec function.

```
GLOBALCONFIG
  ZIIP IPSECURITY
```

Figure 14-1 zIIP IPsec configuration

The other option (the default) is ZIIP NOIPSECURITY, which leaves the IPsec processing running on general CPUs.

Configuring GLOBALCONFIG ZIIP IPSECURITY causes inbound ESP and AH Protocol traffic to be processed in enclave SRBs, and targeted to available zIIPs. Outbound ESP and AH protocol traffic may also be processed on available zIIPs in the following cases:

- ▶ When the application invoking the *send()* function is already running on a zIIP
- ▶ When the data to be transmitted is in response to normal TCP flow control (for example, data transmitted in response to a received TCP acknowledgement or window update)

If the machine does not have a zIIP installed, there is an option to project the CPU in the IEAOPTxx parmlib member called PROJECTCPU. Figure 14-2 shows how to configure this option.

```
PROJECTCPU=YES
```

Figure 14-2 IEAOPTxx PROJECTCPU configuration example

## 14.1.2 Example of zIIP-assisted IPsec implementation

In our test environment we implemented the zIIP-assisted IPsec function by defining a VPN IPsec tunnel between two z/OS images, SC70 and SC65.

### Network configuration

To define a VPN in z/OS is necessary to configure the following components:

- ▶ IPSECURITY in the TCP/IP profile.
- ▶ Policy Agent (PAGENT) address space to handle all the configurations and install them in the TCP/IP stack.
- ▶ Traffic Regulation Management Daemon (TRMD) address space to log all the IPSEC messages.
- ▶ Internet Key Exchange daemon (IKED) address space to perform the key management.
- ▶ SYSLOGD address space to write the messages on a log file.

### IPSECURITY

On a System z9, an additional assist for IPsec protocol traffic is available with the z9 Integrated Information Processor (zIIP). To enable zIIP IP security in Communications Server, specify ZIIP IPSECURITY on the GLOBALCONFIG statement. With zIIP IP security enabled, traffic using the AH and ESP protocols can be processed on available zIIPs. When enabled on a z9 z/OS image that includes zIIPs, the zIIP IP security function can reduce the IPsec processing load on general purpose central processors, beyond what is achievable using just CPACF or the z9 Cryptographic Coprocessor.

When zIIP IP security is enabled, you might need to modify some Workload Manager (WLM) definitions. The IPsec traffic that can be processed on available zIIP processors is assigned to an independent WLM enclave. The WLM independent enclave encapsulates the IPsec

workload as execution units that are separately classified and managed in a WLM service class.

IPSECURITY is configured by using the IPCONFIG statement shown in Figure 14-3.

```
IPCONFIG
IPSECURITY
```

Figure 14-3 IPCONFIG configuration

When IPSECURITY is configured, the TCP/IP automatically installs an implicit *deny all* firewall rule, blocking all the traffic to the stack. In our implementation we defined a rule to allow all the traffic to flow before defining the other rules using the Pagent configuration. VPNs are only supported by using the pagent IPSEC configuration rules. The IPSEC definition in the TCP/IP profile is called the *default policy*.

**Important:** Be careful when defining the IPSECURITY on TCP/IP. The TCP/IP stack has to be restarted or activated to use this function. If the TCP/IP stack is activated with no rules defined, either by the IPSEC statement or by the PAGENT daemon, then all traffic to and from the stack will be blocked.

If you attempt to use the zIIP IPSECURITY support (to direct IPsec AH/ESP protocol processing to zIIP), issue Netstat STATS (or onetstat -S) while IPsec workload is running. The inbound and outbound counters Packets Handled by zIIP will be rising if IPsec workload is in fact being processed on zIIP(s).

If these counters are *not* rising while IPsec traffic is flowing, verify both of the following items:

- ▶ GLOBALCONFIG ZIIP IPSEC parameters are specified in the TCPIP profile (use NETSTAT Config/-f to verify).
- ▶ zIIP(s) are configured to the z/OS image (use MVS D M=CPU command to verify).

## IPSEC statement

The IPSEC statement to define the IPSEC *default policy* is shown in Figure 14-4. This configuration will install the default policy allowing all traffic to flow in and out the stack.

```
IPSEC
IPSECRULE * * NOLOG PROTOCOL *
ENDIPSEC
```

Figure 14-4 IPSEC configuration example

The instructions on how to configure the PAGENT, TRMD, IKED, and SYSLOGD daemon can be found in the following publications:

- ▶ *Communication Server for z/OS V1R7 TCP/IP Implementation, Volume 4: Policy-Based Network Security*, SG24-7169
- ▶ *Sysplex eBusiness Security z/OS V1R7 Update*, SG24-7150
- ▶ *z/OS Communication Server IP Configuration Guide*, SC31-8775
- ▶ *z/OS Communication Server IP Configuration Reference*, SC31-8776

The following definition shows the PAGENT configuration for both systems, SC70 and SC65. Figure 14-5 and Figure 14-6 show the main pagent configuration file. This file points to

configuration files to specific TCP/IP stacks. In our configuration we have only one TCP/IP address space called TCPIP.

```
LogLevel 127
TcpImage TCPIP /u/rodolfi/policy/sc70/tcpip.policy purge flush
```

Figure 14-5 SC70 pagent configuration file referenced by the pagent daemon

```
LogLevel 127
TcpImage TCPIP /u/rodolfi/policy/sc65/tcpip.policy purge flush
```

Figure 14-6 SC65 pagent configuration file referenced by the pagent daemon

Figure 14-7 and Figure 14-8 show the configuration files for the TCPIP stack in both systems. They point to another file which contains the IPSEC policy definitions.

```
TcpImage TCPIP
IPSecConfig /u/rodolfi/policy/sc70/tcpip.policy.ipsec
```

Figure 14-7 SC70 TCPIP stack policy

```
TcpImage TCPIP
IPSecConfig /u/rodolfi/policy/sc65/tcpip.policy.ipsec
```

Figure 14-8 SC65 TCPIP stack policy

Appendix C.1, “IPSEC policy configuration for SC70” on page 500 shows the IPSEC configuration for the SC70 system. It was generated using the IBM Configuration Assistance for z/OS CS V1R9.

Appendix C.2, “IPSEC policy configuration for SC65” on page 503 shows the IPSEC configuration for the SC65 system. It was also generated using the IBM Configuration Assistance for z/OS Communications Server V1R9.

There are only two IPSEC rules defined on each of the preceding examples:

- ▶ One rule that allows all inbound and outbound traffic in the TCP/IP stack. Usually this rule does not exist, but in our example we defined it to facilitate the tests.
- ▶ Another rule that creates a VPN on demand between SC70 and SC65 when any type of traffic occurs. “On demand” means that the VPN will be activated when any traffic matching a specific rule is encountered.

Note that an implicit rule denying all traffic is always created either having the *default policy rules* or the *pagent policy*.

To start the VPN between the systems, we issue a ping command to the other side or generate any kind of traffic between the two systems.

For this implementation we created two REXX programs that are kept in a loop. Each one of the systems, SC70 and SC65, will have a client and a server version talking to each other and sending packets. This programs only send (client) and receive (server) data. This is the only traffic we tested in our implementation.

Figure 14-9 shows the Server REXX program.

```

/* REXX */
parse arg port .
src=Socket('Initialize','ziips')
s=Word(Socket('Socket'),2)
ipaddress=Word(Socket('GetHostId'),2)
src=Socket('Bind',s,'AF_INET' port ipaddress)
src=Socket('Listen',s,10)
src=Socket('Ioctl',s,'FIONBIO','OFF')
say 'ZIIPS: Waiting on' ipaddress port '...'
parse value Socket('Accept',s) with . ns . np nia .
say 'ZIIPS: Connected by' nia 'on port' np 'and socket' ns
call Socket 'Ioctl',ns,'FIONBIO','OFF'
call Time 'R'
bytes=0
do forever
  parse value Socket('Read',ns) with rc size .
  if rc<>0 then leave
  bytes=bytes+size
  if Time('E')>10
  then do
    say 'ziips throughput:' port ipaddress np nia size,
      Format(bytes/Time('E')/1204/1024,5,3)
    if bytes=0 then leave
    bytes=0
    call Time('R')
  end
end
call Socket 'Close',ns
call Socket 'Terminate'
exit

```

Figure 14-9 REXX server

Figure 14-10 on page 207 shows the Client rexx program.

```

/* REXX */
parse arg image port .
src=Socket('Initialize','ziipc')
s=Word(Socket('Socket'),2)
src=Socket('Connect',s,'AF_INET' port '9.12.4.'image)
trace off
nb=64000
data=Copies('*',nb)
call Time 'R'
bytes=0
do forever
  parse value Socket('Write',s,data) with rc nb .
  bytes=bytes+nb
  if Time('E')>10
    then do
      say 'ziipc throughput:' image port nb,
        Format(bytes/Time('E')/1024/1204,5,3)
      bytes=0
      call Time('R')
    end
end
src=Socket('Terminate')
exit

```

Figure 14-10 REXX client

We have to start the client and the server on each side by using the following commands:

```

On SC70:

ziips1 1952 (1952 is the listening port)

On SC65:

ziips1 1952

On SC70:

ziipc1 48 1952 (48 is the last octet of the ip address and 1952 the TCP port)

On SC65:

ziipc1 202 1952

```

Figure 14-11 REXX startup example

After starting the programs, we can monitor the zIIP-assisted IPsec environment and configuration.

First we have to check whether the zIIP is available to be used. We issue the **D M=CPU** command.

```
D M=CPU
IEE174I 13.47.59 DISPLAY M 523
PROCESSOR STATUS
ID CPU          SERIAL
00 +           16991E2094
01 +           16991E2094
02 +A          16991E2094
03 +A          16991E2094
04 +I          16991E2094
05 +I          16991E2094

CPC ND = 002094.S18.IBM.02.00000002991E
CPC SI = 2094.710.IBM.02.00000000002991E
CPC ID = 00
CPC NAME = SCZP101
LP NAME = A16      LP ID = 16
CSS ID = 1
MIF ID = 6

+ ONLINE      - OFFLINE      . DOES NOT EXIST      W WLM-MANAGED
N NOT AVAILABLE

A      APPLICATION ASSIST PROCESSOR (zAAP)
I      INTEGRATED INFORMATION PROCESSOR (zIIP)
CPC ND CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID CENTRAL PROCESSING COMPLEX IDENTIFIER
CPC NAME CENTRAL PROCESSING COMPLEX NAME
LP NAME LOGICAL PARTITION NAME
LP ID   LOGICAL PARTITION IDENTIFIER
CSS ID  CHANNEL SUBSYSTEM IDENTIFIER
MIF ID  MULTIPLE IMAGE FACILITY IMAGE IDENTIFIER
```

Figure 14-12 D M=CPU command

As you can see in Figure 14-12, we have two zIIPs available in the system (CPU ID 04 and 05), and they are online.



The `netstat -S` command in Figure 14-13 shows how many packets are being handled in a zIIP processor.

```
RODOLFI @ SC65:/u/rodolffi>netstat -S
MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIP          14:31:01
IP Statistics
Packets Received                = 1095389266
Inbound Calls from Device Layer = 0
Inbound Frame Unpacking Errors  = 0
Inbound Discards Memory Shortage = 0
Received Header Errors          = 0
Received Address Errors         = 22
Datagrams Forwarded            = 1
Unknown Protocols Received      = 0
Received Packets Discarded      = 0
Received Packets Delivered      = 1095282062
Output Requests                 = 1170732490
Output Discards No Route        = 216
Output Discards DLC Sync Errors  = 0
Output Discards DLC Async Errors = 1
Output Discards Memory Shortage = 0
Output Discards (other)         = 0
Reassembly Timeouts             = 0
Reassembly Required             = 0
Reassembly Successful           = 0
Reassembly Failures             = 0
Datagrams Successfully Fragmented = 0
Datagrams Failing Fragmentation = 0
Fragments Created               = 0
Fragments Created               = 0
Inbound Packets handled by zIIP = 1074645575
Outbound Packets handled by zIIP = 873127703
```

Figure 14-13 Netstat -S command example

The `netstat -f` (configuration) command in Figure 14-14 can be used to check whether the zIIP support is enabled.

```
RODOLFI @ SC65:/u/rodolffi>netstat -f
MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIP          14:37:00
Global Configuration Information:
TcpIpStats: No  ECSALimit: 0000000K PoolLimit: 0000000K
MlsChkTerm: No  XCFGRPID:      IQDVLANID: 0
SegOffload: No  SysplexWLMPoll: 060
ExplicitBindPortRange: 00000-00000
Sysplex Monitor:
TimerSecs: 0060 Recovery: Yes DelayJoin: No AutoRejoin: No
MonIntf: No DynRoute: No
zIIP:
IPSecurity:Yes
```

Figure 14-14 Netstat -f command example

The Resource Measurement Facility (RMF) monitor can be used to check how much CPU is being used by the zIIP. In our test we used the Monitor III to show some details of how the zIIP is being utilized by LPARs and address spaces.

The CPC Capacity screen in Monitor III shows how the CPU is being used by partitions; see Figure 14-15.

```

RMF V1R9   CPC Capacity                               Line 1 of 61

Samples: 119   System: SC65   Date: 05/16/07   Time: 12.04.00   Range: 120   Sec

Partition:  A11           2094 Model 710
CPC Capacity:  640   Weight % of Max: ****   4h Avg:  29   Group:  N/A
Image Capacity: 256   WLM Capping %:  0.0   4h Max:  52   Limit:  N/A

Partition  --- MSU ---  Cap Proc   Logical Util %   - Physical Util % -
              Def Act  Def  Num    Effect  Total   LPAR Effect Total

*CP
A11          0   37  NO   2.0    28.2  28.8   0.1   5.6   5.8
A16          0   33  NO   2.0    24.8  25.5   0.1   5.0   5.1
*IIP
A11          NO   2.0    39.2  39.5   0.3   39.2  39.5
A16          NO   2.0    37.7  38.0   0.3   37.7  38.0

```

Figure 14-15 RMF CPC Capacity example

Figure 14-15 displays how the partitions are using zIIP, A16 (SC70), and A11 (SC65).

The next RMF examples, Figure 14-16 and Figure 14-17 on page 211, show how the zIIP is being used by the address spaces on SC70 and SC65 systems in the RMF Processor Usage screen.

```

RMF V1R9   Processor Usage                               Line 1 of 11

Samples: 119   System: SC70   Date: 05/16/07   Time: 12.04.00   Range: 120   Sec

Jobname  CX  Service  --- Time on CP % ---  ----- EAppl % -----
              Class  Total  AAP  IIP      CP  AAP  IIP

RODOLFI3 0  SYSSTC1  24.4  0.0  0.0    24.4  0.0  0.0
RODOLFI4 0  SYSSTC1   8.9  0.0  0.0     8.9  0.0  0.0
TCP/IP    SO  SYSSTC   4.8  0.0  0.0     4.8  0.0  74.0
RMFGAT   SO  SYSSTC   1.5  0.0  0.0     1.5  0.0  0.0
XCFAS    S  SYSTEM   0.7  0.0  0.0     0.7  0.0  0.0
WLM      S  SYSTEM   0.6  0.0  0.0     0.6  0.0  0.0
VTAM44   S  SYSSTC   0.2  0.0  0.0     0.2  0.0  0.0
*MASTER* S  SYSTEM   0.1  0.0  0.0     0.1  0.0  0.0
SMSVSAM  S  SYSTEM   0.1  0.0  0.0     0.1  0.0  0.0
JES2     S  SYSSTC   0.1  0.0  0.0     0.1  0.0  0.0
CATALOG  S  SYSTEM   0.1  0.0  0.0     0.1  0.0  0.0

```

Figure 14-16 SC70 RMF Processor Usage example

As we can see in Figure 14-16, the TCP/IP address space is using most of the CPU. It is responsible for the ESP and AH protocol processing, encrypting and decrypting all the data. Figure 14-17 on page 211 shows the SC65 system and we can see that most of the CPU utilization was displaced to the zIIPs.

RMF V1R9 Processor Usage		Line 1 of 16								
Samples: 119		System: SC65		Date: 05/16/07		Time: 12.04.00		Range: 120		Sec
Jobname	CX	Service Class	--- Time on CP % ---			----- EApp1 % -----				
			Total	AAP	IIP	CP	AAP	IIP		
RODOLFI3	0	SYSSTC1	28.7	0.0	0.0	28.7	0.0	0.0		
RODOLFI4	0	SYSSTC1	8.7	0.0	0.0	8.7	0.0	0.0		
TCPIP	SO	SYSSTC	5.5	0.0	0.0	5.5	0.0	76.9		
RMFGAT	SO	SYSSTC	1.4	0.0	0.0	1.4	0.0	0.0		
XCFAS	S	SYSTEM	0.8	0.0	0.0	0.8	0.0	0.0		
WLM	S	SYSTEM	0.6	0.0	0.0	0.6	0.0	0.0		
AOPMSG	SO	STC	0.4	0.0	0.0	0.4	0.0	0.0		
VTAM44	S	SYSSTC	0.3	0.0	0.0	0.3	0.0	0.0		
HZSPROC	SO	STC	0.3	0.0	0.0	0.3	0.0	0.0		
VAINI	T	TSO	0.3	0.0	0.0	0.3	0.0	0.0		
*MASTER*	S	SYSTEM	0.1	0.0	0.0	0.1	0.0	0.0		
GRS	S	SYSTEM	0.1	0.0	0.0	0.1	0.0	0.0		
SMSVSAM	S	SYSTEM	0.1	0.0	0.0	0.1	0.0	0.0		
IXGLOGR	S	SYSTEM	0.1	0.0	0.0	0.1	0.0	0.0		
CATALOG	S	SYSTEM	0.1	0.0	0.0	0.1	0.0	0.0		
GPMSERVE	SO	STC	0.1	0.0	0.0	0.1	0.0	0.0		

Figure 14-17 SC65 RMF Processor Usage example

It is possible to enable and disable the zIIP usage by using the OBEYFILE command. We just have to change the option in GLOBALCONFIG to ZIIP NOIPSECURITY and update the profile to disable the zIIP utilization. To enable, simply configure back to ZIIP IPSECURITY. Now we can turn the zIIP off on SC65 and see how the system will behave, as shown in Figure 14-18.

RMF V1R9 Processor Usage		Line 1 of 11								
Samples: 120		System: SC65		Date: 05/16/07		Time: 12.21.00		Range: 120		Sec
Jobname	CX	Service Class	--- Time on CP % ---			----- EApp1 % -----				
			Total	AAP	IIP	CP	AAP	IIP		
TCPIP	SO	SYSSTC	46.1	0.0	0.0	46.1	0.0	0.0		
RODOLFI4	0	SYSSTC1	30.7	0.0	0.0	30.7	0.0	0.0		
RODOLFI3	0	SYSSTC1	15.7	0.0	0.0	15.7	0.0	0.0		
RMFGAT	SO	SYSSTC	1.3	0.0	0.0	1.3	0.0	0.0		
XCFAS	S	SYSTEM	0.6	0.0	0.0	0.6	0.0	0.0		
WLM	S	SYSTEM	0.6	0.0	0.0	0.6	0.0	0.0		
RMF	S	SYSSTC	0.5	0.0	0.0	0.5	0.0	0.0		
AOPMSG	SO	STC	0.4	0.0	0.0	0.4	0.0	0.0		
VTAM44	S	SYSSTC	0.2	0.0	0.0	0.2	0.0	0.0		
*MASTER*	S	SYSTEM	0.1	0.0	0.0	0.1	0.0	0.0		
GPMSERVE	SO	STC	0.1	0.0	0.0	0.1	0.0	0.0		

Figure 14-18 SC65 RMF Processor Usage example 2

The jobname TCPIP represents the TCP/IP address space and the jobnames rodolf3 and rodolfi4 are the REXX server and client programs, respectively. Without the zIIP, the CPU utilization was displaced back to the general CPs. For the CPC activity screen, see the Figure 14-19 on page 212.

RMF V1R9		CPC Capacity		Line 1 of 61						
Samples:	120	System:	SC65	Date:	05/16/07	Time:	12.21.00	Range:	120	Sec
Partition:	A11	2094 Model 710								
CPC Capacity:	640	Weight % of Max:	****	4h Avg:	31	Group:	N/A			
Image Capacity:	256	WLM Capping %:	0.0	4h Max:	57	Limit:	N/A			
Partition	---	MSU	---	Cap	Proc	Logical Util %	- Physical Util % -			
	Def	Act	Def	Num	Effect	Total	LPAR	Effect	Total	
*CP				66.0			5.6	68.0	73.6	
A11	0	66	NO	2.0	51.5	51.7	0.0	10.3	10.3	
A16	0	26	NO	2.0	19.4	20.0	0.1	3.9	4.0	
*IIP				12.0			1.3	32.5	33.8	
A11			NO	2.0	0.0	0.0	0.0	0.0	0.0	
A16			NO	2.0	32.4	32.6	0.2	32.4	32.6	

Figure 14-19 RMF CPC Capacity example 2

A white paper called “Capacity Planning for zIIP-assisted IPsec” is available at the following link. It describes in detail how to plan for zIIP capacity.

<http://www.ibm.com/support/docview.wss?rs=852&uid=swg27009459>

## 14.2 Policy-based routing

When TCP/IP has to route a packet to the network, the decision is based on the destination address in the packet IP header. All traffic sent to a destination address has to use the same route. You can have more than one network interface being used to the same route (multipath route); for example, having two OSA cards connected to the same network.

Because the System z9 has the capability of having multiple interfaces, running many different workloads, with different requirements. The only way to accomplish that today is running the application in another partition, or creating another TCP/IP stack with different route tables. In one z/OS image, it is possible to have up to eight TCP/IP stacks, which allows the possibility of having up to eight route tables.

This new policy-based routing allows the packet route to be selected based on one or more of the following criteria:

- ▶ Source IP address
- ▶ Destination IP address
- ▶ Source TCP port
- ▶ Destination TCP port
- ▶ Protocol (UDP and TCP)
- ▶ Jobname
- ▶ Netaccess security zone
- ▶ Multi-level security (MLS) label

The routing decision is always taken for outbound traffic (data being sent from z/OS). The outbound traffic that meets any subset of the criteria listed can be targeted to a specific network interface and first-hop routers.

Now the TCP/IP can have multiple routing tables:

- ▶ At least one, the main routing table, defined in the TCP/IP profile
- ▶ Zero (0) or more policy-based routing tables

A policy-based routing table works the same way as the main routing table. It can contain:

- ▶ Static routes only.
- ▶ Dynamic routes only.
- ▶ A combination of static and dynamic routes.
- ▶ The static routes can be replaceable or non-replaceable (replaceable or not by using dynamic routing via OMPROUTE).

The routes in a policy-based route (PBR) table is limited to specific link or next hops by:

- ▶ Static routes: by defining static routes for the route table using only specific links and first-hops.
- ▶ Dynamic routes: by configuring the links and first-hops to be considered by OMPROUTE. OMPROUTE will only generate routes for the links and first-hops that were specified in the routing table.

PBR is implemented using a pagent policy. The PBR is a new policy in the pagent configuration file. A PBR policy can be created based on a subset of the various criteria.

Traffic that matches a certain criteria, like the jobname for example, can be defined to use up to eight different route tables, plus the main routing table, optionally, as a backup. All traffic that does not match any defined criteria will use the main routing table, if it is defined.

Look at the example in Figure 14-20.

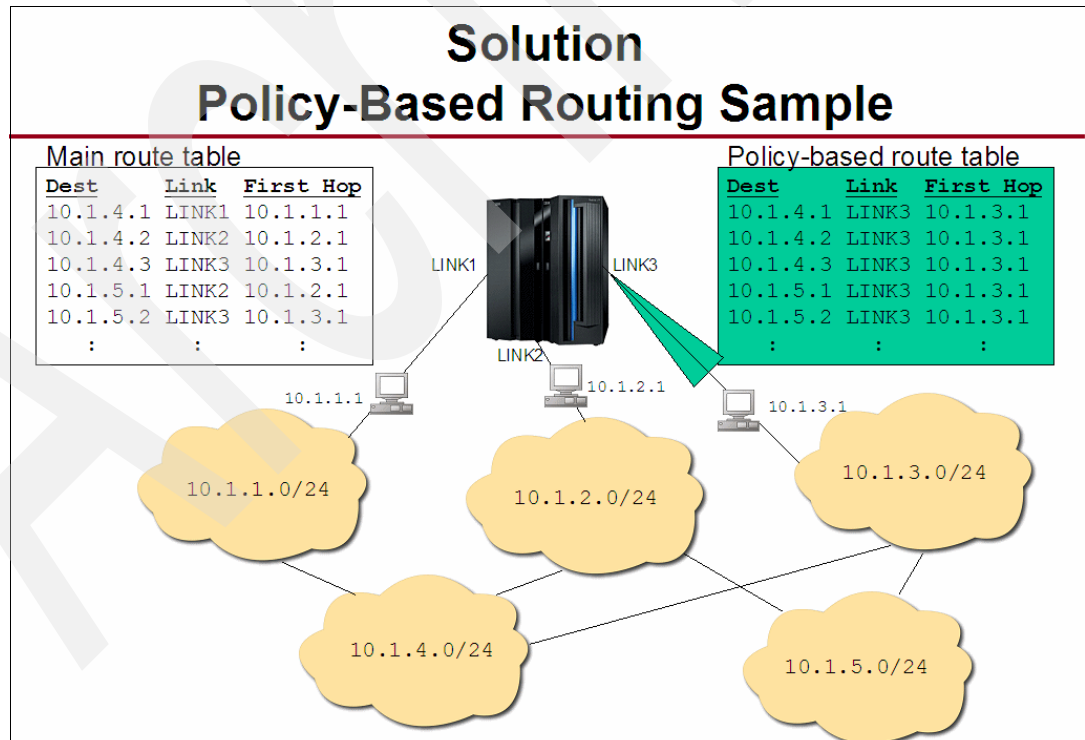


Figure 14-20 Routing scenario example

In the example we have two different routing tables:

- ▶ The main routing table spreading the traffic between three different links: LINK1, LINK2, and LINK3.
- ▶ A PBR policy redirecting all the traffic to a specific link, LINK3. Any criteria could be specified in the PBR policy to utilize this PBR table, like a specific jobname.

The route table is selected as follows:

1. Each route table defined for a specific traffic is searched, in the order they are defined, for a route to the traffic destination.
2. If any active route to the destination (from the more specific to more general, host, subnet, network, supernet, or default) matched the packet destination, then that route is used.
3. The main route table will be searched last, if it defined to be used as a backup.

The routing selection algorithm performed for a single PBR table is the same as the algorithm used for the main route table.

**Note:** The routing selection algorithm works the following way:

1. If a route exists to the destination address, a host route, then it is used.
2. If no host route exists to the destination address, then:
  - a. If subnet, network or supernet routes exists to the destination, then the route with the most specific network mask (the mask with most of the bits on) will be used.
  - b. If the destination is a multicast destination and a multicast default route exists, then that route is used.
3. Default routes are used when no other route exists to a destination.

The only type of traffic supported by the PBR policy is the locally originated IPv4 TCP and UPD. The main route table will always be used for:

- ▶ All IPv6 traffic
- ▶ All forwarded traffic
- ▶ All traffic using IP protocols other than TCP and UDP, including ICMP (ICMP Echo requests, PING, will continue to be routed using the main route table).

## 14.2.1 Policy-based routing implementation

The PBR policy is configured in a pagent flat-file. It consists of:

- ▶ Routing rules
- ▶ Routing actions
- ▶ Routing tables

The LDAP is not supported for the PBR policy, only flat-files are supported. The IBM Configuration Assistant for z/OS Communication Server can be used to generate the PBR configuration flat file. In our implementation we demonstrate how to create a simple PBR policy using the configuration assistant.

### Routing rules

A routing rule specify a set of traffic characteristics and an action to be taken for outbound traffic that matches those characteristics.

A routing rule definition consist of:

- ▶ Source IP address:
- ▶ Destination IP address
- ▶ Traffic descriptor (characteristics)
- ▶ Priority
- ▶ Time condition
- ▶ A reference to a routing action

The source and the destination IP address can be specified as a single IP address, a prefix/mask specification, a range of IP addresses, or it can reference a group of IP addresses.

**Important:** The source IP address for an outbound TCP connection or an outbound UDP packet can be influenced by a number of configuration and application options. See the source IP address information in *z/OS Communications Server: IP Configuration Guide* for the hierarchy of ways that the source IP address of an outbound packet is determined.

For the following source IP address selection methods, a route lookup is needed to determine the source IP address:

- ▶ SOURCEVIPA: static VIPA address from the HOME list
- ▶ HOME IP address of the link over which the packet is sent

Do not use the IP source address as a selector for traffic that relies on one of the above two methods to select its source IP address. At the time the route lookup is done, the source IP address is not known.

The traffic descriptor specifies the following characteristics of outbound traffic:

- ▶ Source port, single or a range of ports
- ▶ Destination port, single or a range of ports
- ▶ TCP or UDP protocol
- ▶ Job name: a trailing asterisk (\*) can be used as a wild card
- ▶ Security zone: netaccess security zone
- ▶ Security label: Multi-level security

The traffic descriptor can be specified inline, as a reference, or as a reference to a group of traffic descriptors.

The priority controls the order in which rules are searched for a match. Some rules for priority specification:

- ▶ Can be specified from 1 to 2.000.000.000 (2 billion)
- ▶ Default is 1 (lowest priority)
- ▶ If a packet can match more than one rule, priority should be used to ensure that the rules are searched in the intended order.
- ▶ For rules with the same priority, the order in which the rules are searched is unpredictable.

**Note:** Priority is not explicitly configured when the Configuration Assistant is used to generate the routing configuration file. The rule priority is determined by the order of the rules as shown on the rules panel.

The time condition specifies when the routing rule is to be active. It is possible to specify a specific date, a date range, a mask for months of the year and days of the week.

## Routing action

Routing action specifies the routing tables to be used for traffic that matches a routing rule. Up to eight routing tables can be specified for a routing action, and they are searched in the order specified.

There is an option to use or not the main route table, specified in the TCP/IP profile, if no active route is found in any of the specified policy-based route tables.

## Route table

The route table defines a policy-based route table. The maximum number of policy-based route tables that can be defined for a TCP/IP stack is 255. Only active route tables are installed in the TCP/IP stack. Route tables are considered active if it is referenced by an active routing rule and its associated action.

A policy-based route table can contain static routes, dynamic routes or both (IPv4 routes only). A route table definition consists of:

- ▶ Table name
- ▶ Route entries (static routes)
- ▶ Dynamic routing parameters entries (controls the calculation of dynamic routes by OMPROUTE)
- ▶ Advanced parameters

The route table name uniquely identify a policy-based route table and has to be 1 to 8 characters long. The name EZBMAIN and ALL in upper case, lower case, or mixed case are reserved. The EZBMAIN is the main route table, defined in the TCP/IP profile. We recommend you define all the route table names either all in upper case or all in lower case.

A route entry defines a static route and the syntax is similar to that of BEGINROUTES statement in TCP/IP profile.

The dynamic routing parameters are used by OMPROUTE to control the dynamic routes added to the policy-based route table. Multiple dynamic routing parameters can be configured on a route table. Each dynamic routing table parameter consists of a link name and a first hop IP address.

The advanced parameters are:

- ▶ Multipath: the main route table uses the multipath definition in TCP/IP profile. If a different multipath definition is needed, the options are:
  - Use global (as defined in the IPCONFIG statement in TCP/IP profile)
  - Per connection
  - Per packet
  - Disable (use only the first active route to a destination)
- ▶ Ignore path mtu update: indicates whether IPv4 ICMP Fragmentation Needed messages should be ignored for this route table.
- ▶ Dynamic XCF routes: indicates whether direct routes to dynamic XCF addresses on other TCP/IP stacks should be added to this route table

For a complete description of the PBR implementation and statements syntax, refer to the following publications:

- ▶ *SC31-8775 - z/OS V1R9 Communication Server IP Configuration Guide*
- ▶ *SC31-8776 - z/OS V1R9 Communication Server IP Configuration Reference*



**Important:** Considerations on FLUSH/NOFLUSH and PURGE/NOPURGE policy agent options for policy-based routing:

- ▶ FLUSH/NOFLUSH:
  - NOFLUSH option is not supported.
  - Routing policies are always deleted prior to installing new policies at the following times:
    - Policy agent startup.
    - Tcplmage/PEPInstance statement added.
    - MODIFY/REFRESH command issued.
- ▶ PURGE/NOPURGE:
  - PURGE option is not supported.
  - Routing policies are never deleted during policy agent shutdown or when a Tcplmage/PEPInstance statement is deleted.

To remove all routing policies from a TCP/IP stack, delete the RoutingConfig statement from the policy agent image configuration file for the stack.

## 14.2.2 Policy-based routing implementation example

Here we describe how to implement a simple PBR policy using the configuration assistant. Keep these considerations in mind when using the configuration assistant for PBR policy definitions:

- ▶ Only stack-specific routing policy files are created (no common routing policy).
- ▶ Use the RoutingConfig statement in pagent configuration file to specify the routing policy file name.
- ▶ Routing rules are called “connectivity rules” within the Configuration Assistant.
- ▶ The routing action is combined with the routing rule as a single object to configure.
- ▶ The routing rule priority value is not manually configured. Priority is set based on the order of rules as displayed on rules panel.
- ▶ The policy agent configuration statements that are generated will be read by policy agent with no syntax errors.

Now start the configuration assistance. Figure 14-21 on page 218 shows the first screen.

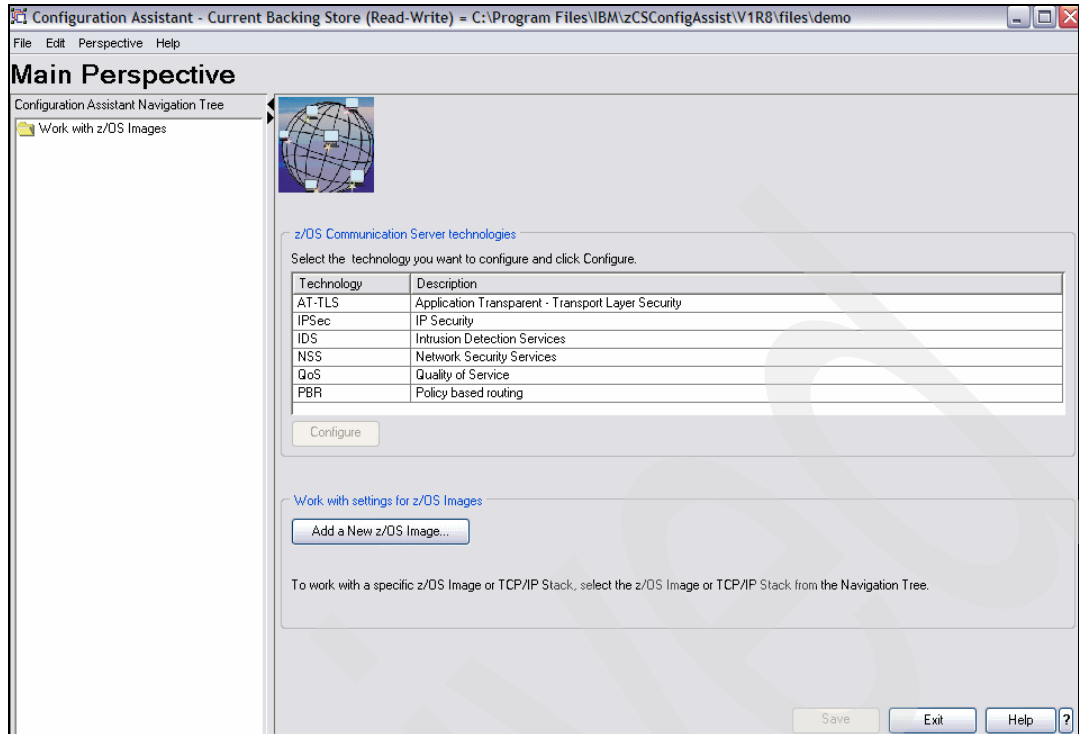


Figure 14-21 Configuration assistance main screen

The configuration assistance in z/OS 1.9 has some improvements from the previous release. Previously, for a particular policy like AT\_TLS or IPSEC, configuration assistance stored the policies configuration in different files, one for each policy. Now, all the policies are kept in a single configuration file.

In the main screen we have now different technologies, called *perspectives*, for each possible policy.

First we created the z/OS images, SC65 and SC70. With the right mouse button, we clicked **Work with z/OS images** then selected **add new z/OS image**. Figure 14-22 shows the screen where you enter the name of the image.

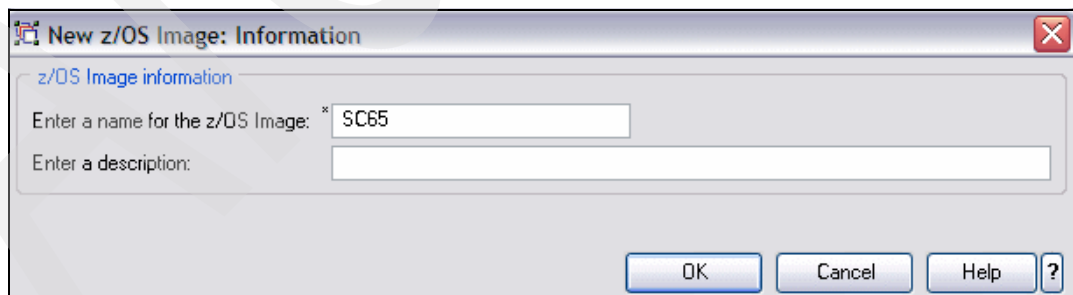


Figure 14-22 Configuration assistance image creation

We clicked **OK** to create the image. Then we repeated the same steps to create the SC70 image. After inserting an image, a question popped up asking if we wanted to add a stack for the image we created. We answered No. After creating the images, the following screen displayed.

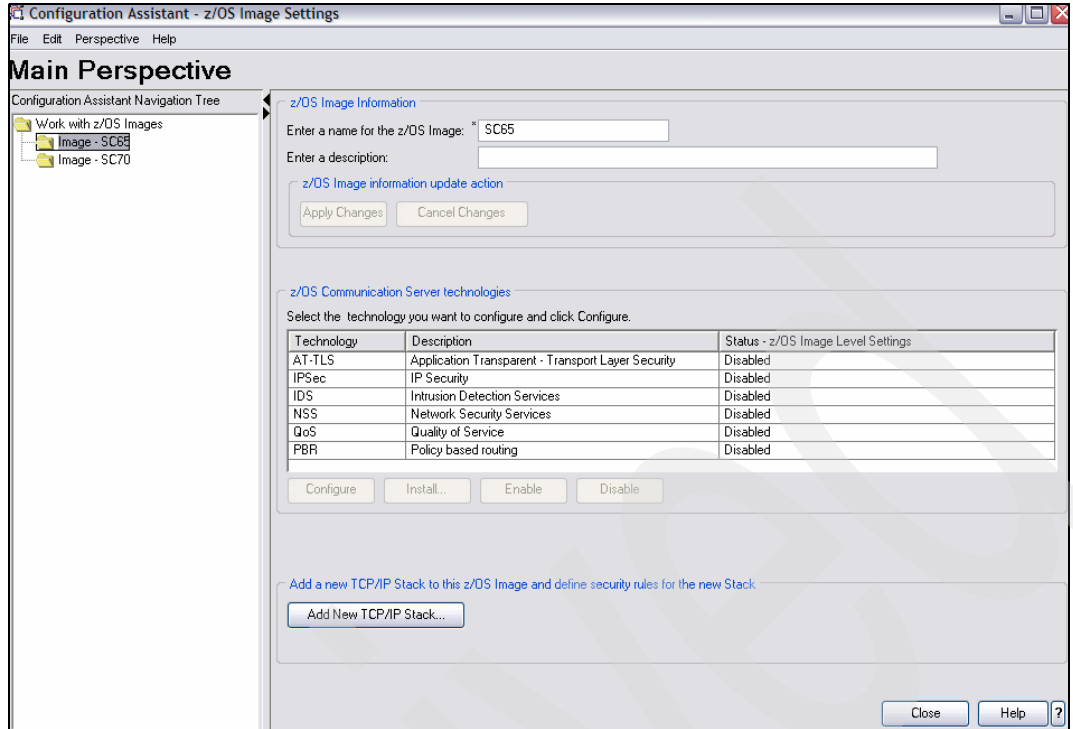


Figure 14-23 Configuration assistance images created

Now we have to enable the policies we want to work with. We selected the PBR policy and enabled it by clicking **Enable**. Then we clicked **Add New TCP/IP Stack**, and Figure 14-24 displayed.

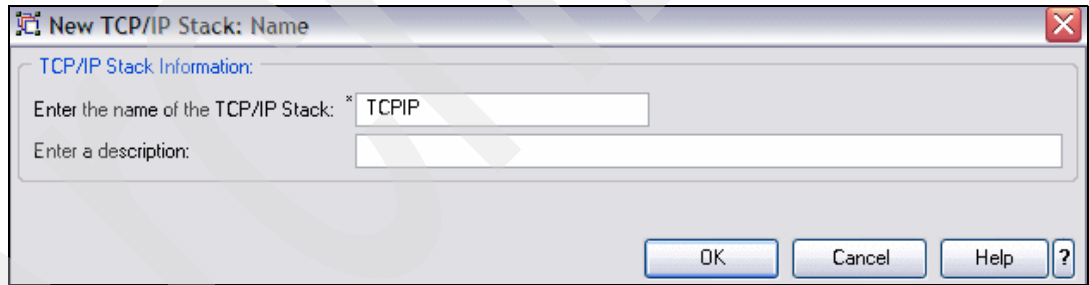


Figure 14-24 Configuration assistance new stack

After adding the stack, TCPIP, for both systems, we had to enable the PBR policy again for the stack on every image. We selected the stack on one of the images as shown in Figure 14-25 on page 220, then selected the PBR policy and clicked **Enable**.

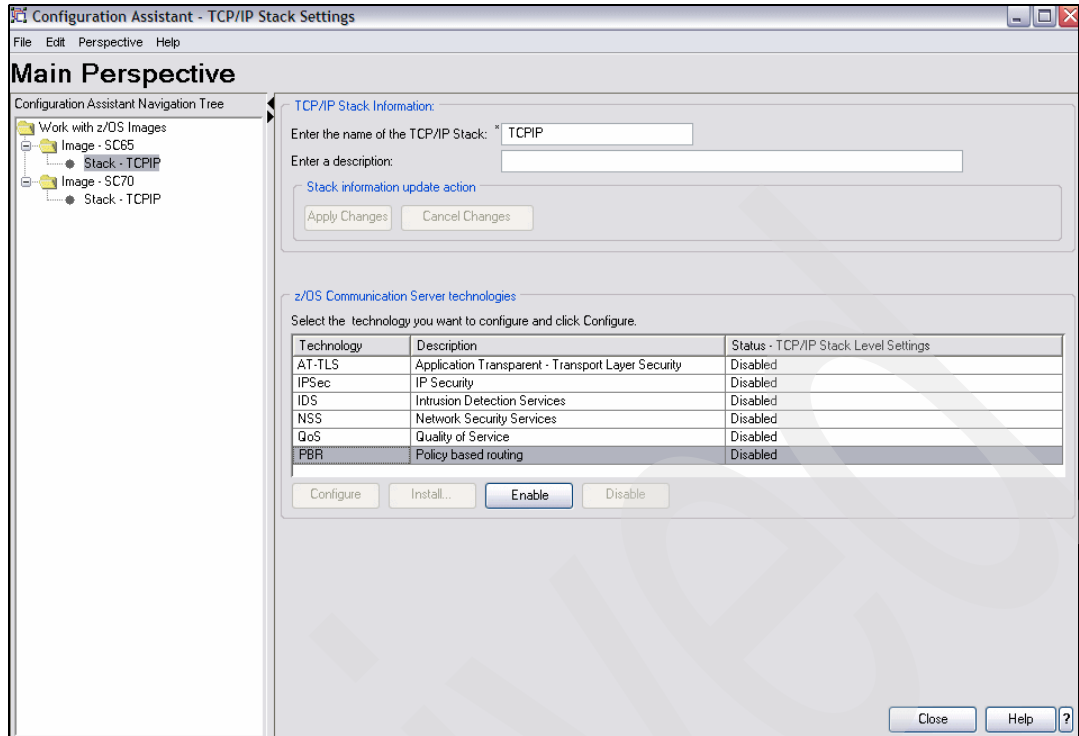


Figure 14-25 Configuration assistance stack policy enablement

The status of the policy will appear as Incomplete, in red, because there is no configuration defined for the stack; see Figure 14-26.

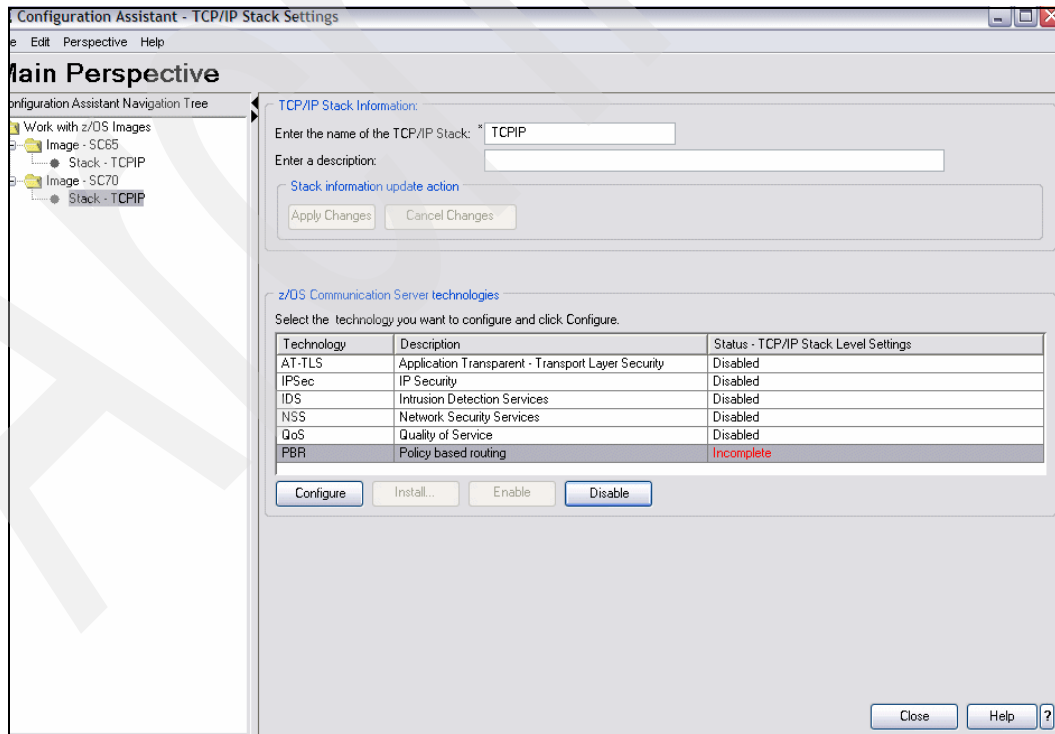


Figure 14-26 Configuration assistance stack policy enablement 2

We clicked **Configure**. The configuration assistance asked if we wanted to configure a connectivity rule; we clicked Yes and reached Figure 14-27.

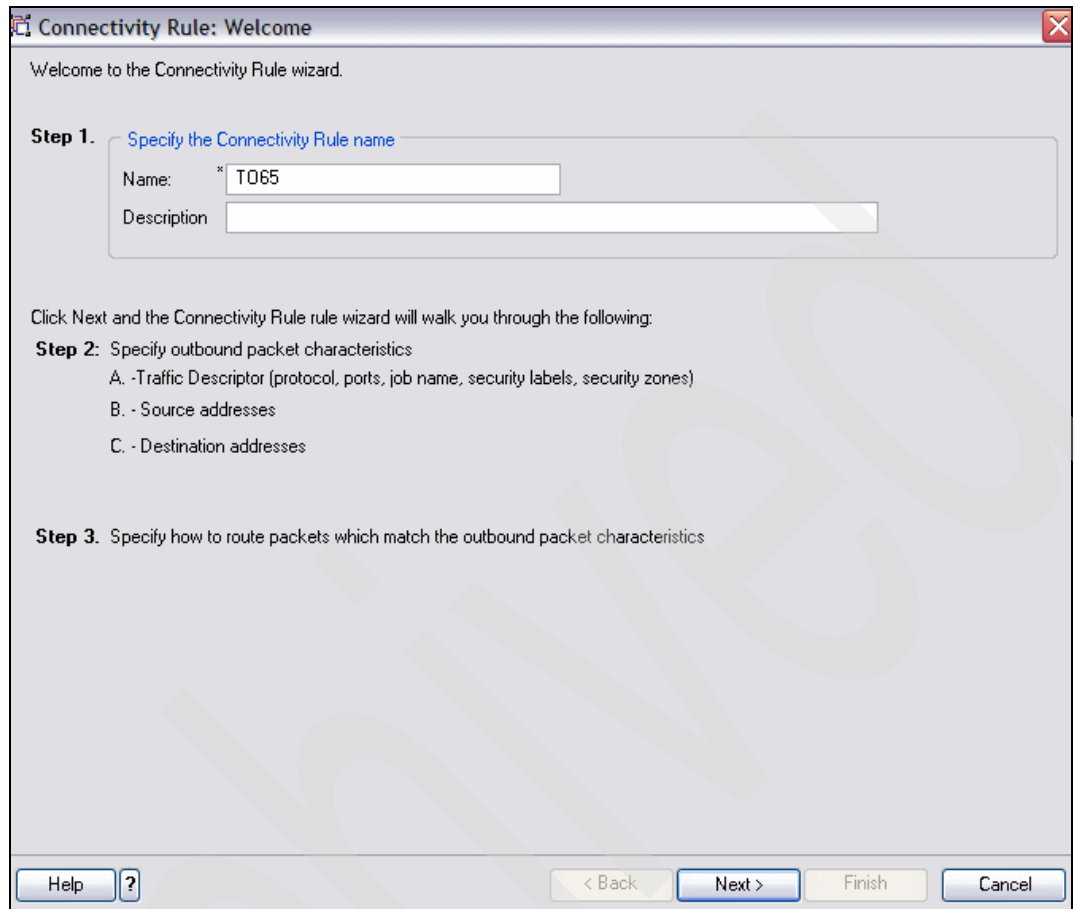


Figure 14-27 Configuration assistance rule name

We choose the name T065 for this rule, and then clicked **Next**. Now we had to define a traffic descriptor to be used for that rule. In our case we used the same REXX programs from the zIIP IPsec implementation. We added a TCP traffic descriptor as shown in Figure 14-28 on page 222.

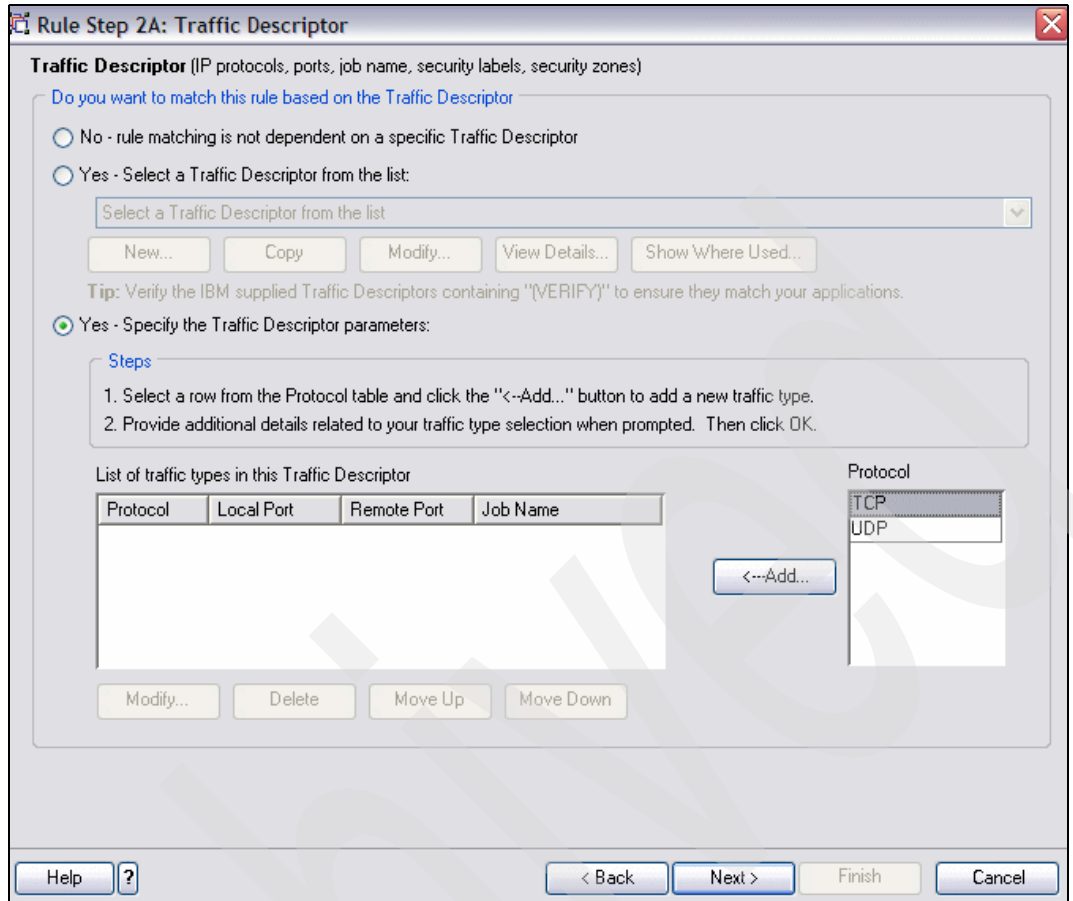


Figure 14-28 Configuration assistance traffic descriptor creation

We selected **Yes - Specify the Traffic Descriptor parameters:**, and then clicked **Add**. Figure 14-29 displayed, showing the characteristics of the traffic we wanted.

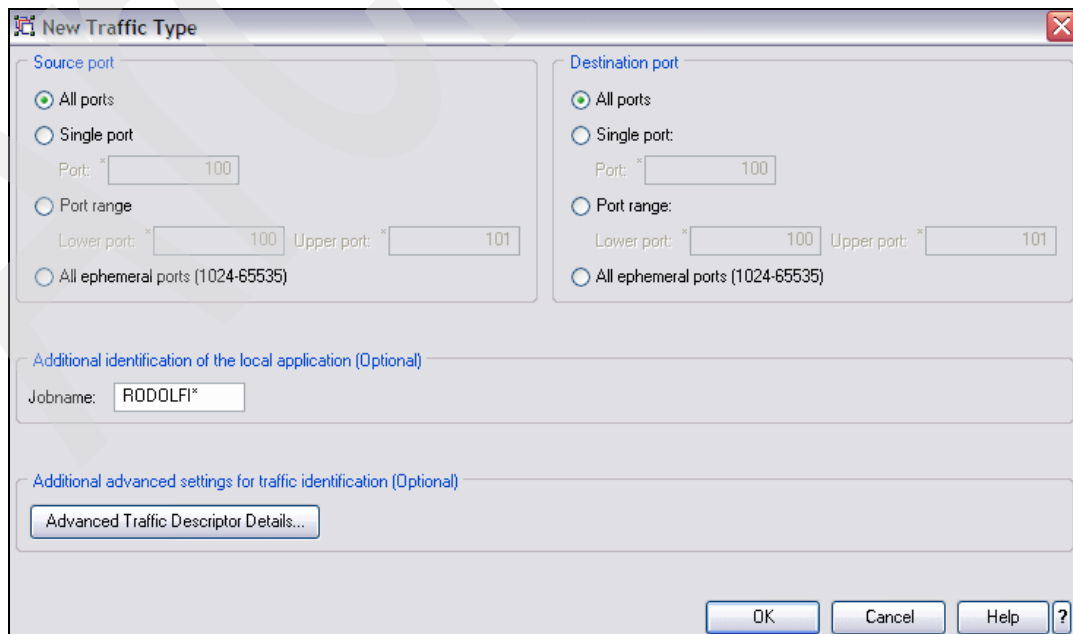


Figure 14-29 Configuration assistance new traffic type

For Source, we selected **All ports**, for Destination we selected **All ports** and all jobnames initiating by RODOLFI\*, then clicked **OK**. Figure 14-30 then displayed.

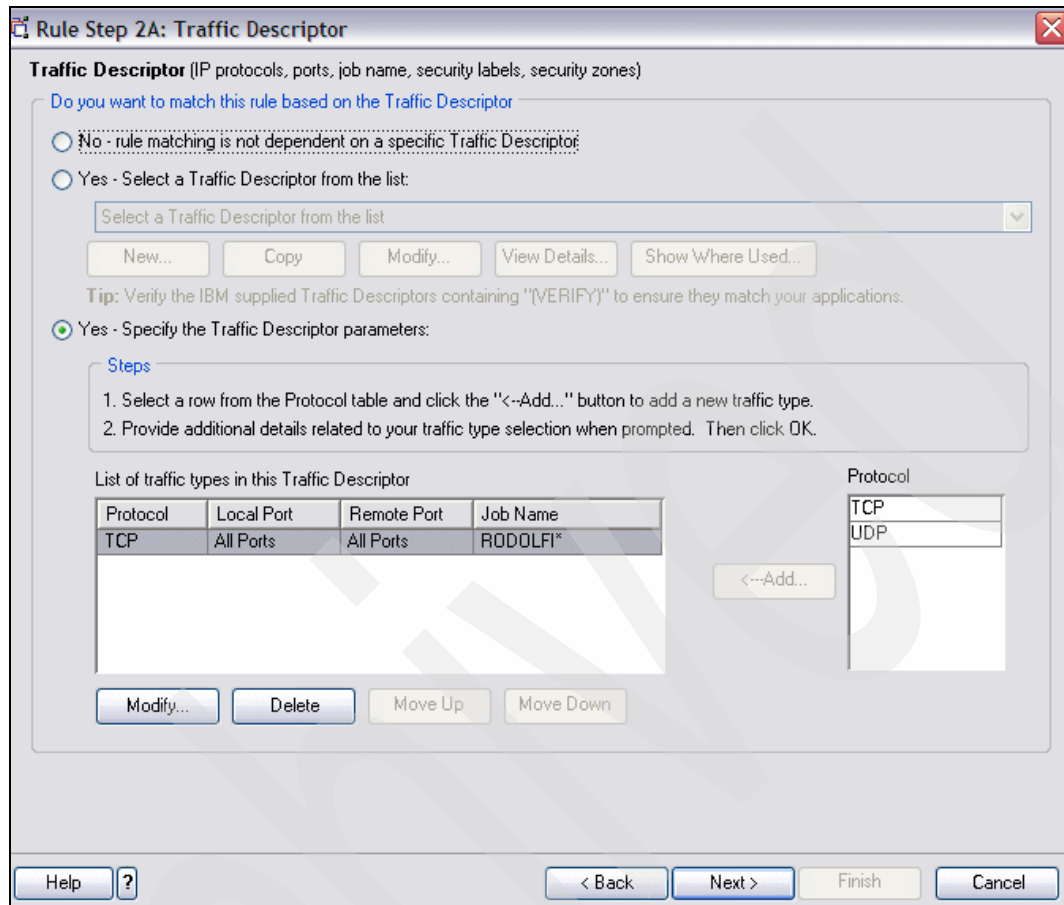


Figure 14-30 Configuration assistance traffic descriptor defined

Now the traffic descriptor is defined. We clicked **Next**. Now we specified an IP source address (in our case, 9.12.4.202) for the SC70 image; see Figure 14-31 on page 224.

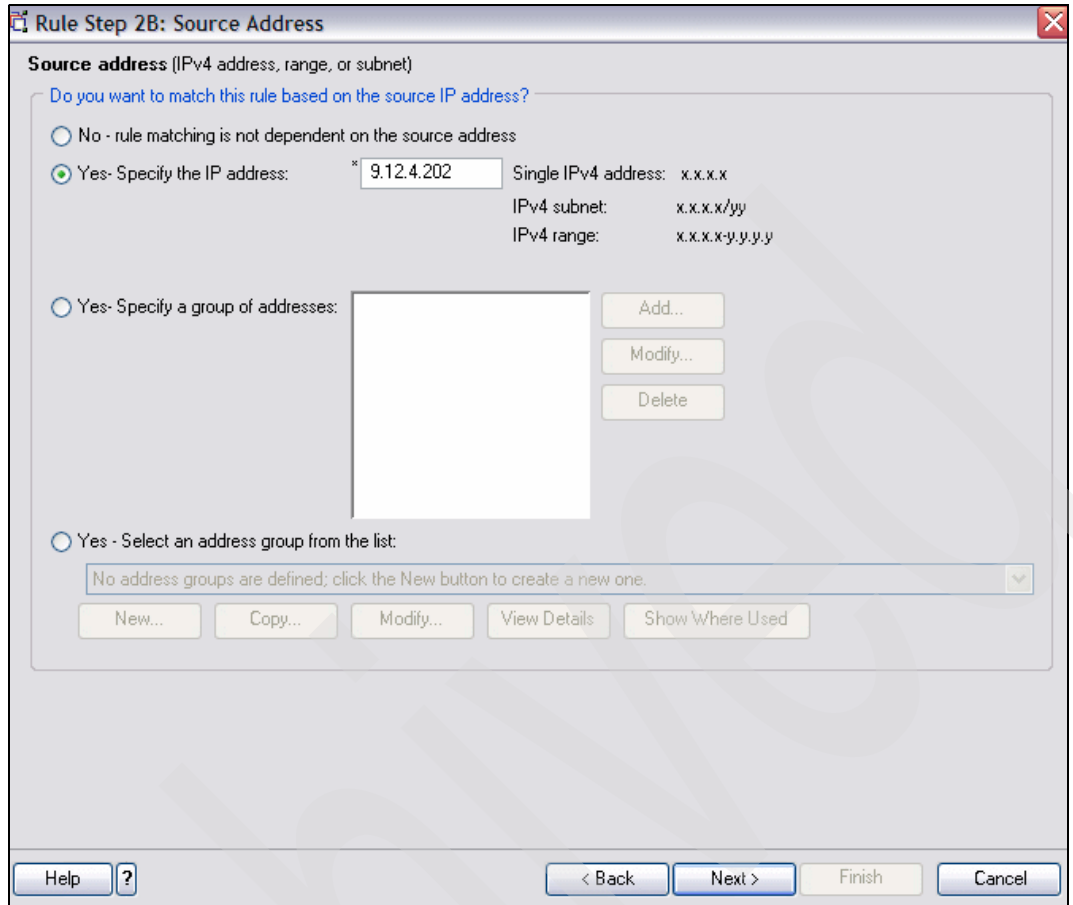


Figure 14-31 Configuration assistance source ip address

Then we clicked **Next** to configure the destination IP address, 9.12.4.48, as shown in Figure 14-32 on page 225.



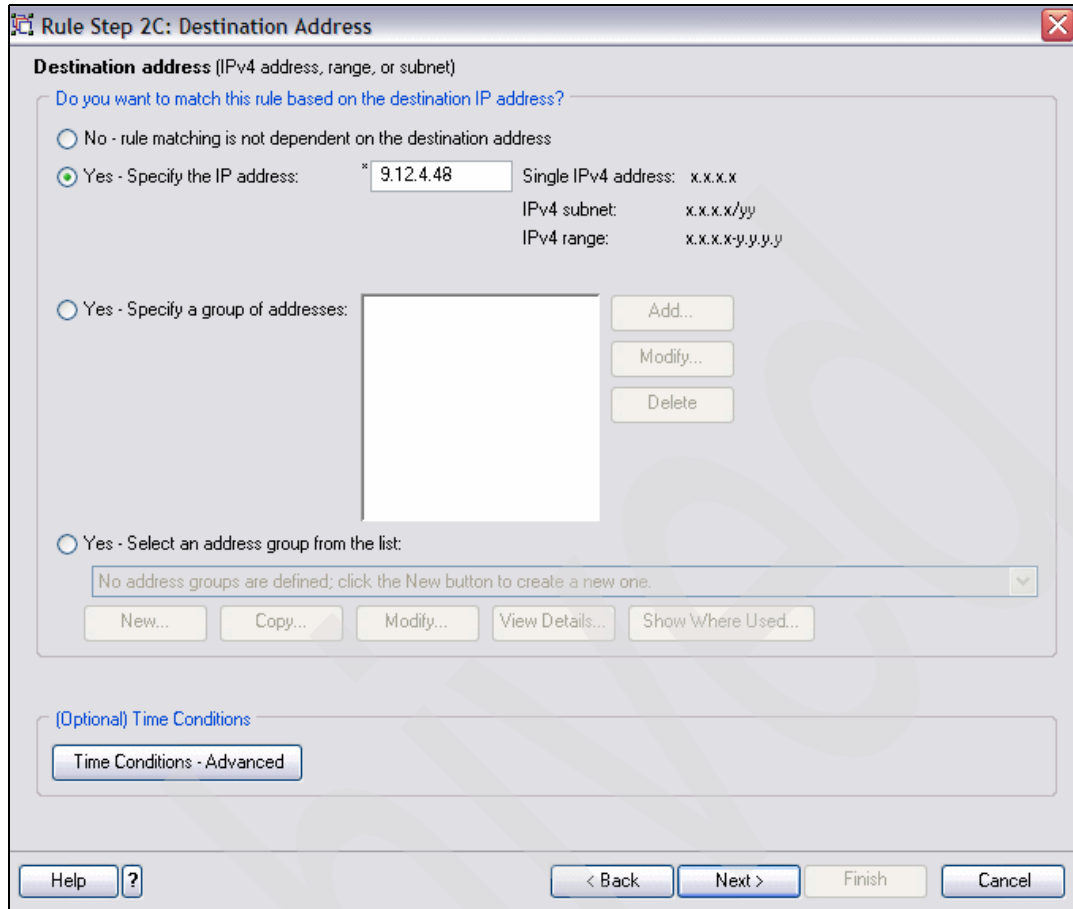


Figure 14-32 Configuration assistance destination IP address

Then we clicked **Next** to define a route table, as shown in Figure 14-33 on page 226.

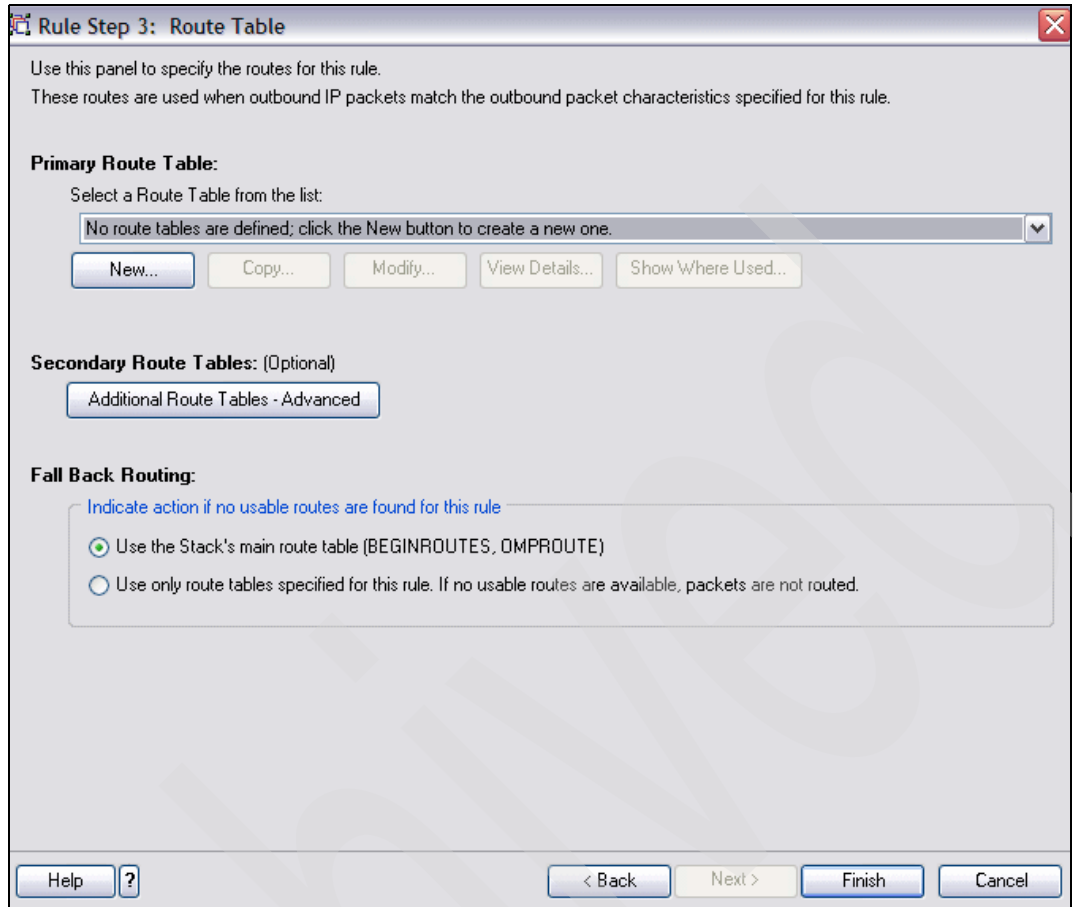


Figure 14-33 Configuration assistance new route table

We created a new route table. As an example, we defined a new route redirecting the traffic for the two partitions from the OSA card to the XCF links. In our configuration we had the iQDIO defined. All traffic flowing between these two partitions will be done by using the iQDIO links. See the main routing tables for both partitions, SC65 and SC70, in Figure 14-34 and Figure 14-35 on page 227.

```

RODOLFI @ SC65:/u/rodolfi>netstat -r
MVS TCP/IP NETSTAT CS V1R9      TCP/IP Name: TCP/IP      16:32:22
Destination      Gateway          Flags      Refcnt  Interface
-----
Default          9.12.4.1       UGS       000000  OSA2020LNK
9.12.4.0/22      0.0.0.0        US        000003  OSA2020LNK
9.12.4.48/32    0.0.0.0        UH        000000  OSA2020LNK
9.12.4.49/32    0.0.0.0        UH        000000  STAVIPA1LNK
10.1.101.0/24    0.0.0.0        US        000000  IQDIOLNK0A016541
10.1.101.63/32  0.0.0.0        UHS       000000  IQDIOLNK0A016541
10.1.101.64/32  0.0.0.0        UHS       000000  IQDIOLNK0A016541
10.1.101.65/32  0.0.0.0        H         000000  EZASAMEMVS
10.1.101.65/32  0.0.0.0        UH        000000  IQDIOLNK0A016541
10.1.101.70/32  0.0.0.0        UHS       000000  IQDIOLNK0A016541
127.0.0.1/32    0.0.0.0        UH        000000  LOOPBACK

```

Figure 14-34 SC65 main routing table

```

RODOLFI @ SC70:/u/rodolfi>netstat -r
MVS TCP/IP NETSTAT CS V1R9          TCPIP Name: TCPIP          16:30:58
Destination      Gateway          Flags      Refcnt  Interface
-----
Default          9.12.4.1        UGS        000000  OSA2020LNK
9.12.4.0/22      0.0.0.0         US         000003  OSA2020LNK
9.12.4.202/32    0.0.0.0         UH         000000  OSA2020LNK
9.12.4.203/32    0.0.0.0         UH         000000  STAVIPA1LNK
10.1.101.0/24    0.0.0.0         US         000000  IQDIOLNKO016546
10.1.101.63/32   0.0.0.0         UHS        000000  IQDIOLNKO016546
10.1.101.64/32   0.0.0.0         UHS        000000  IQDIOLNKO016546
10.1.101.65/32   0.0.0.0         UHS        000000  IQDIOLNKO016546
10.1.101.70/32   0.0.0.0         H          000000  EZASAMEMVS
10.1.101.70/32   0.0.0.0         UH         000000  IQDIOLNKO016546
127.0.0.1/32     0.0.0.0         UH         000000  LOOPBACK

```

Figure 14-35 SC70 main routing table

We redirected the traffic from the OSA card, OSA2020 link, to the iQDIO links, IQDIOLNNK\*. Next we created a new route table. We clicked **New**, as shown in Figure 14-33 on page 226. Then Figure 14-36 on page 228 was shown.

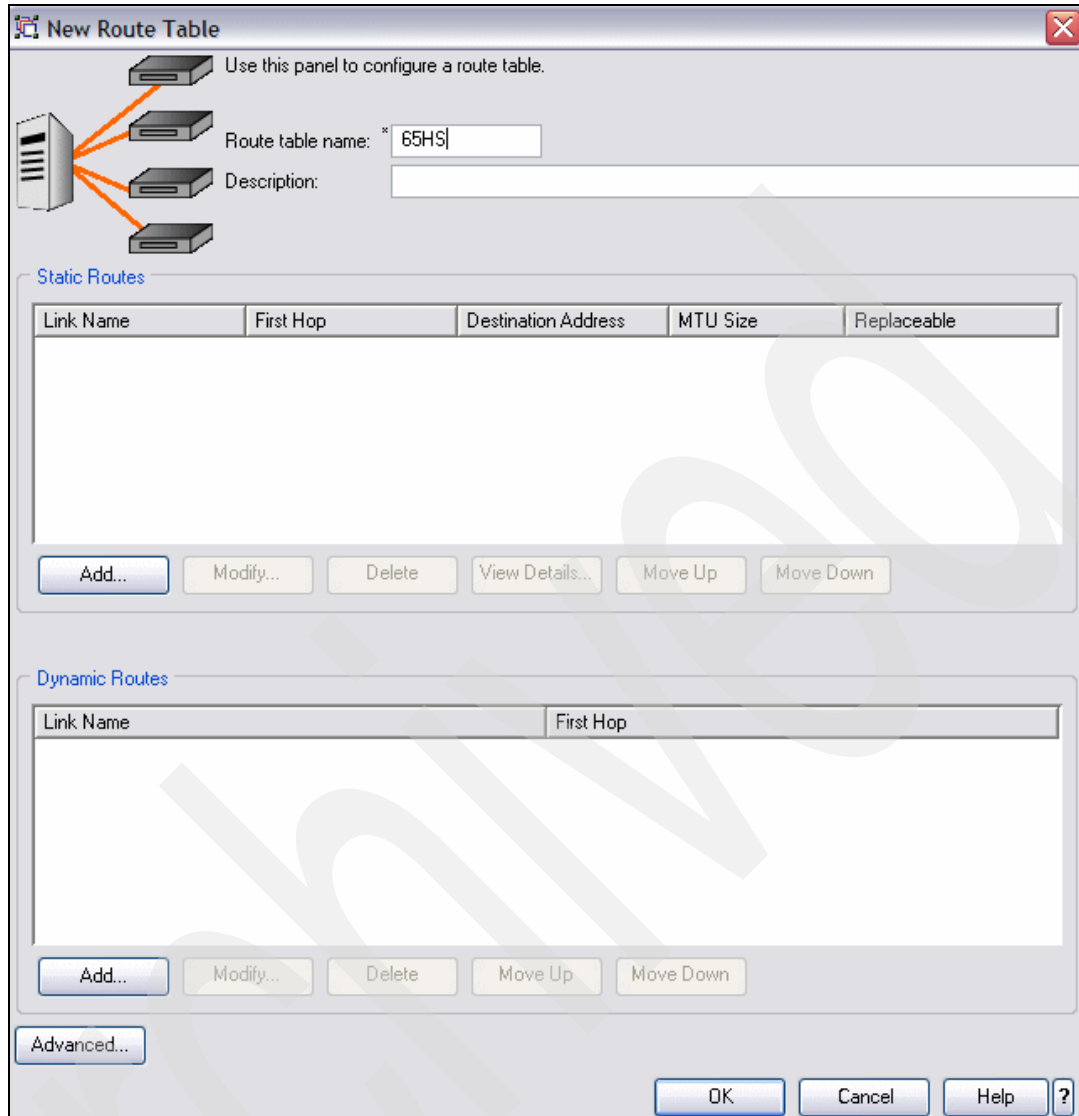


Figure 14-36 Configuration assistance adding a route table

We choose 65HS (65 via HiperSockets™) as the route table name. Then we added a new static rule by clicking **Add** under the static routes list (empty now). Figure 14-37 on page 229 was displayed.

**New Static Route Table Entry**

Destination address: \* 9.12.4.48      DEFAULT      any IPv4 address  
 IPv4 address: x.x.x.x  
 IPv4 subnet: x.x.x.x/yy

First hop address: \* 10.1.101.65      DIRECT      first hop equals destination  
 IPv4 address: x.x.x.x

Link name: \* IQDIOLNK0A016

MTU size: \* 1500 (bytes)

Allow this route entry to be replaced by OMPROUTE

Advanced Settings

OK      Cancel      Help      ?

Figure 14-37 Configuration assistance new rule parameters

The static rule will contain the following parameters:

- ▶ Destination address: 9.12.4.48 (OSA address for partition SC65)
- ▶ First hop address: 12.1.101.65 (XCF link address for partition SC65)
- ▶ Link name: IQDIOLNK0A016546 (the link name for the IQDIO)
- ▶ MTU size: 1500

Then we clicked **OK** to create this static rule. Figure 14-38 on page 230 displayed.

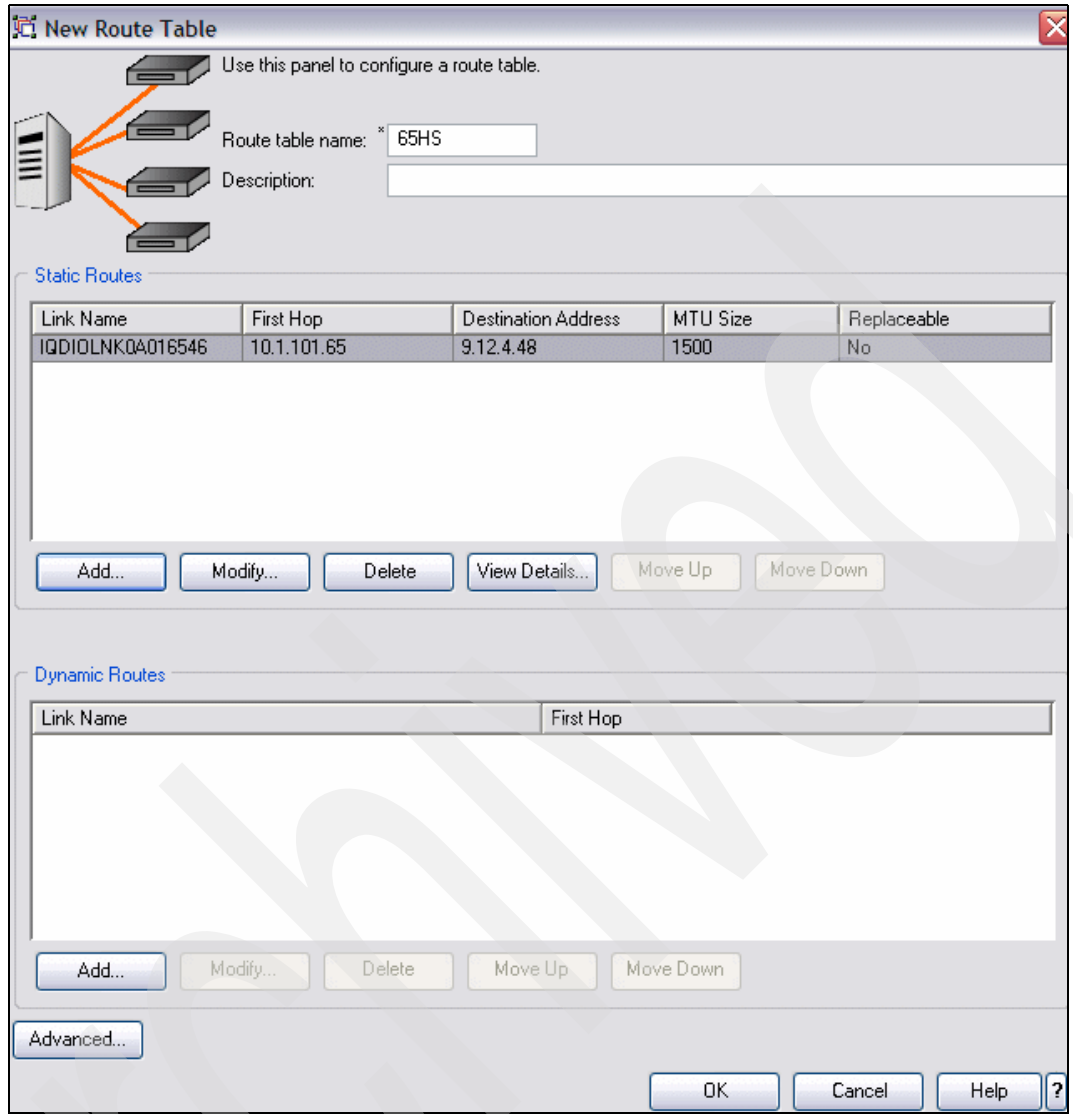


Figure 14-38 Configuration assistance new route table created

Then we clicked **OK** to complete the process of creating a PBR rule. Figure 14-39 on page 231 displayed.

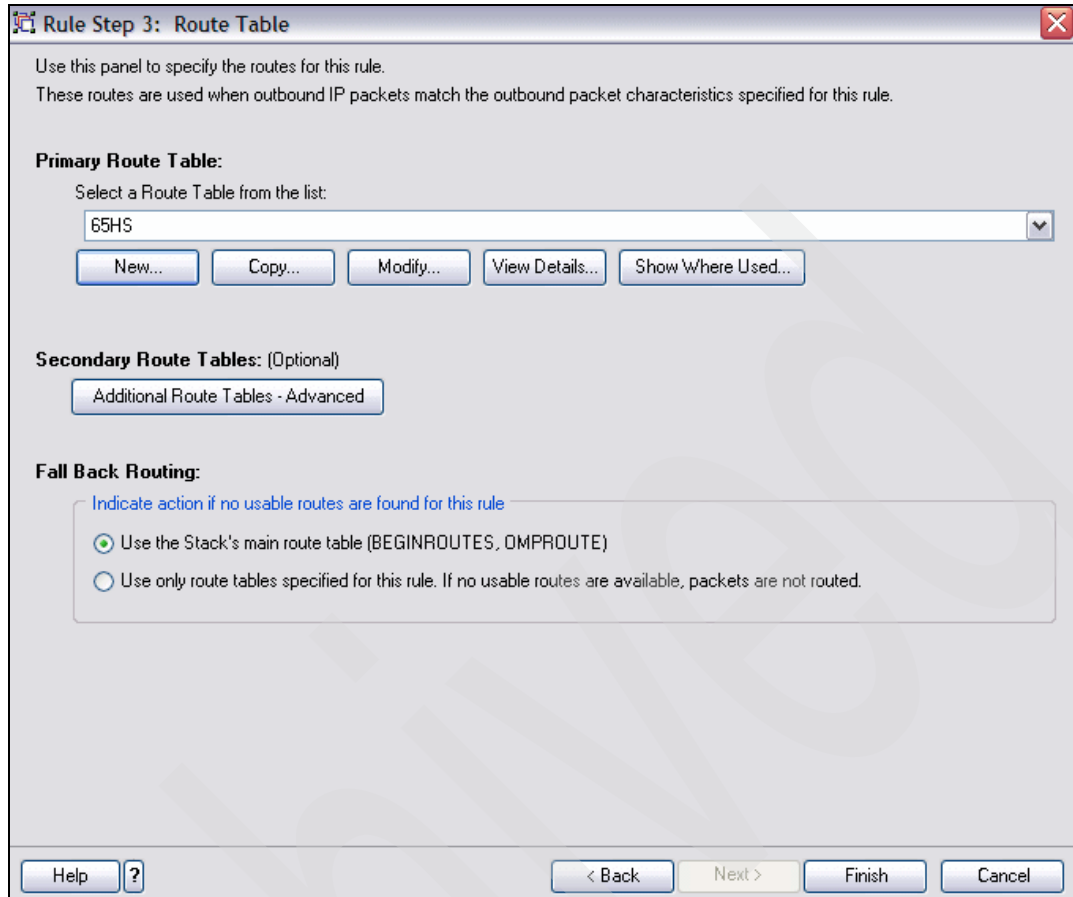


Figure 14-39 Configuration assistance finishing the pbr rule

Then we clicked **Finish** and the SC70 definitions were created. Figure 14-40 on page 232 displayed.

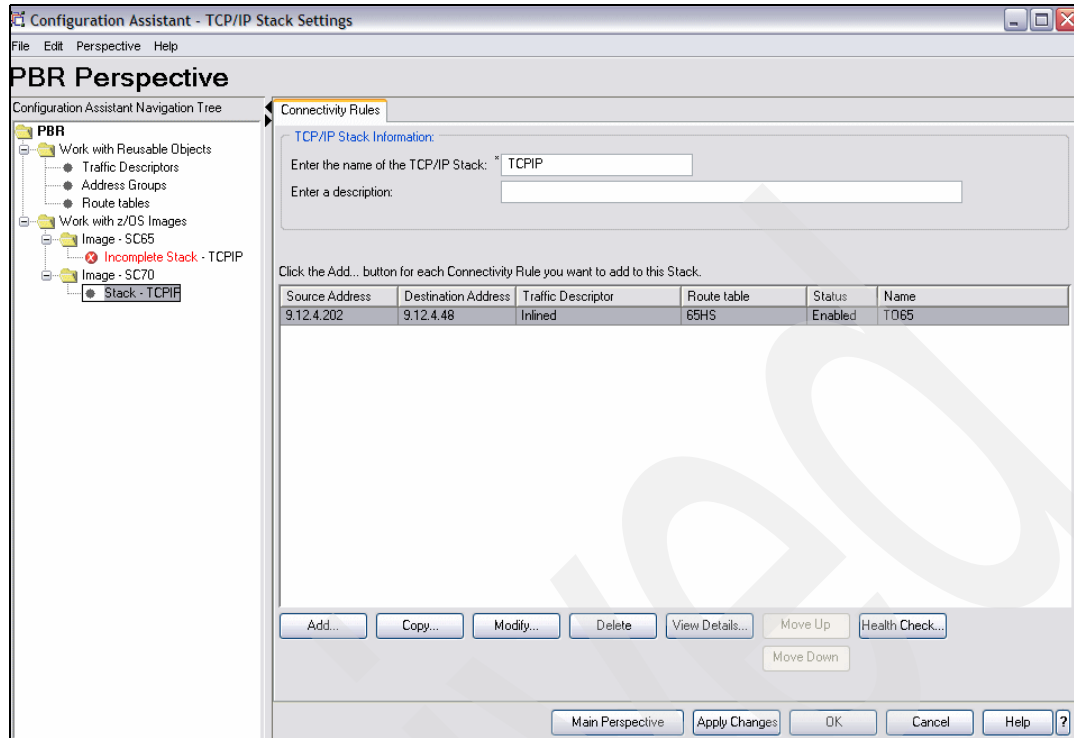


Figure 14-40 Configuration assistance SC70 pbr rule created

We clicked **Apply Changes** to save the configuration under the image. Then we repeated the same steps for the system SC65 using the following parameters:

- ▶ Connectivity rule name: TO70
- ▶ Same traffic descriptor
- ▶ IP source address: 9.12.4.48
- ▶ IP destination address: 9.12.4.202
- ▶ Route table name: 70HS
- ▶ Static rule entry:
  - Destination address: 9.12.4.48
  - First hop address; 10.1.101.70
  - Link name: IQDIOLNK0A016541
  - MTU size: 1500

Now we had both systems configured by the configuration assistance. Appendix C.3, “SC65 pbr configuration files” on page 507 and Appendix C.4, “SC70 pbr configuration files” on page 507 show how the PBR configuration file will look for both systems.

Next we had to install all the policies on both systems by:

- ▶ Transferring the files using the configuration assistance or any other FTP client
- ▶ Updating the pagent configuration

To transfer the policies to the systems, we clicked one of the images in the left panel, as shown in Figure 14-41 on page 233.



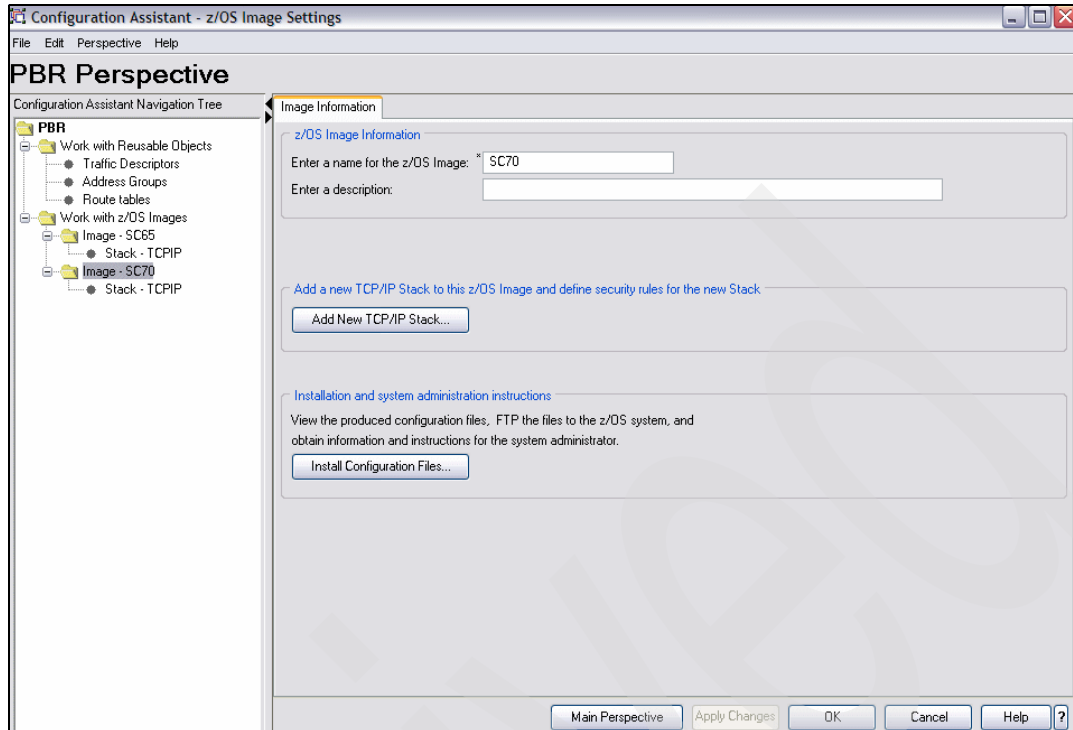


Figure 14-41 Configuration assistance transferring the files

Then we clicked **Install Configuration Files**, and the following screen displayed.

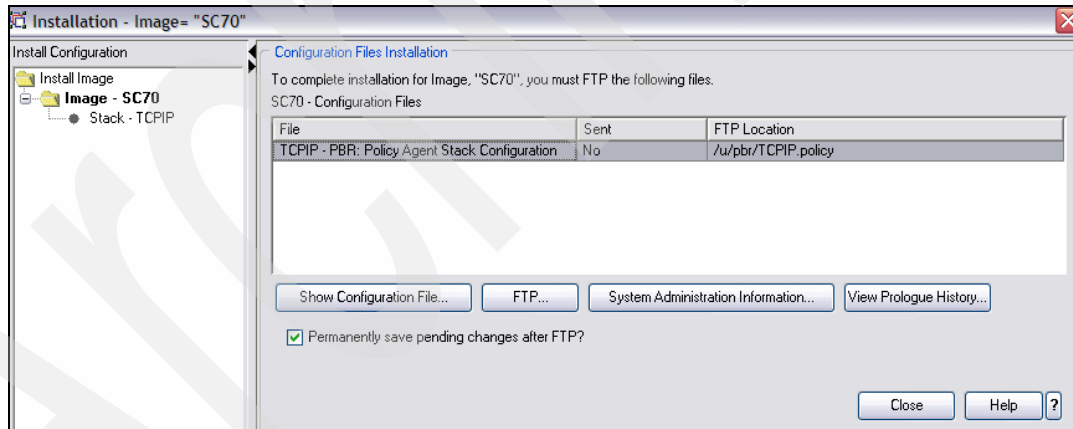


Figure 14-42 Configuration assistance file transfer options

We selected the file to transfer and clicked **FTP**. Note that the Sent information displays No. That means the FTP policy configuration was changed in that file and has not transferred to z/OS yet. Each time we perform a change in any configuration that affects a configuration file, this field will display No, indicating that we have to transfer the file to z/OS. After clicking **FTP**, Figure 14-43 on page 234 will display.

Figure 14-43 Configuration assistance FTP parameters

We completed the FTP information to send the file and then clicked **Send** to transfer the configuration file to the appropriate system and location.

After the file is successfully transferred, you need to update the pagent configuration file to point to the PBR configuration and update the configuration by using the modify command on z/OS console.

The new pagent configuration file for both systems are shown in the following figures.

```

RODOLFI @ SC70:/u/rodolfi/policy/sc70>cat tcpip.policy
TcpImage TCPIP
IPSecConfig /u/rodolfi/policy/sc70/tcpip.policy.ipsec
RoutingConfig /u/rodolfi/policy/sc70/tcpip.policy.pbr

```

Figure 14-44 SC70 pagent configuration file for stack tcpip

```
RODOLFI @ SC65:/u/rodolfi/policy/sc65>cat tcpip.policy
TcpImage TCPIP
IPSecConfig /u/rodolfi/policy/sc65/tcpip.policy.ipsec
RoutingConfig /u/rodolfi/policy/sc65/tcpip.policy.pbr
```

Figure 14-45 SC70 pagent configuration file for stack tcpip

Next, we activated the new configuration by using the **modify update** command on the pagent address space.

```
F PAGENT65,UPDATE
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : ROUTING
```

Figure 14-46 Pagent update command

To verify if the new policies are being used in both systems, you can issue the command **netstat -A**. This command shows for each connection whether a PBR policy is being used; refer to the examples in Appendix C.5, “SC65 netstat -A command” on page 508 and Appendix C.6, “SC70 netstat -A command” on page 510.

In both systems, for the connections established by the REXX programs, the PBR policy is being used. The **pasearch** command was also changed to support the new PBR policy. The **-R** option was added to display the policies; refer to the example in Appendix C.7, “pasearch -R command” on page 512.

Archived

## System REXX for z/OS

System REXX for z/OS Base is part of the simplification trend introduced with the “New Face of z/OS”. SYSREXX (or System REXX) is to provide an infrastructure through which REXX execs may be run outside the normal TSO/E or batch environments, using a simple programming interface. This enables the leveraging of base operating system components by new style applications that will, over time, lead to simplified interaction and more intuitive system management capabilities on z/OS.

The possibilities for exploiting existing REXX code through the use of SYSREXX are vast, whether to provide operator assists or to provide routines that can be leveraged by new strategic initiatives. System REXX provides a gateway for new style applications to interface with z/OS components

While the New Face matures, System REXX adds real value today through exploitation possibilities for operator assists and also certain system management processes.

## 15.1 Introduction to System REXX (SYSREXX)

System REXX is a new base element in z/OS V1R9. As a new part of the BCP, System REXX includes an addition to the Capacity Provisioning component (FMID HPV7740) and the System REXX for z/OS base component. System REXX for z/OS base was a Web deliverable in z/OS V1R8, known as System REXX Support for z/OS V1R8 and z/OS.e V1R8, and was identified by FMID HBB77SR.

**Note:** System REXX is a new facility shipped in z/OS V1R9 and was available via a Web deliverable in z/OS V1R8. It runs REXX execs in an authorized environment, and these REXX execs can invoke TSO/E commands. This new support is made available with a TSO/E APAR OA20186 that enables this capability for z/OS V1R8 systems.

The introduction of System REXX in z/OS is due to a requirement to provide an infrastructure to support Web-based interactions with z/OS components as part of the “New Face of z/OS” initiative for simplifying z/OS management. The cornerstone of this new infrastructure is SYSREXX, which allows execs to be run simply and independently from traditional TSO/E and batch environments. This implementation has two interfaces:

- ▶ A single program interface (AXREXX)
- ▶ Operator exploitation directly from a console

The expected benefit of this implementation is to:

- ▶ Enable rapid development and deployment of system programmer tools and operator assists
- ▶ Be exploited by new and old style applications
- ▶ Allow health checks to be written in REXX

As a beginning in z/OS V1R9, SYSREXX is the required environment for CIM and Health Checker REXX execs to be written.

## 15.2 SYSREXX address space (AXR)

AXR is a subsystem that is started during master subsystem initialization. It reads the AXR00 parmlib member and allocates the REXXIN data set. When REXX work arrives via a PC directly into the appropriate server, SYSREXX or console-initiated REXX execs are detected by the AXR SSI listener, converted to F AXR command format, and queued to the command server's CIB control block. Then, in turn they are selected and scheduled for processing.

### Removing the AXR address space

The System REXX address space, AXR, is non-cancelable, but can be terminated by invoking the following command:

```
FORCE AXR,ARM
```

When the AXR address space terminates, an ENF signal 65 with a qualifier of 40000000x is issued. AXR can be restarted by starting the AXRPSTRT procedure, which can be found in SYS1.PROCLIB. When the AXR address space initializes, an ENF signal of 80000000x is issued. To restart the AXR address space, issue:

```
S AXRPSTRT
```

## 15.2.1 SYSREXX from consoles

When REXX work requests originate from an operator console, that is detected by an AXR SSI listener function (shown in Figure 15-1), or a program interface called using the AXREXX macro service (shown in Figure 15-2 on page 240).

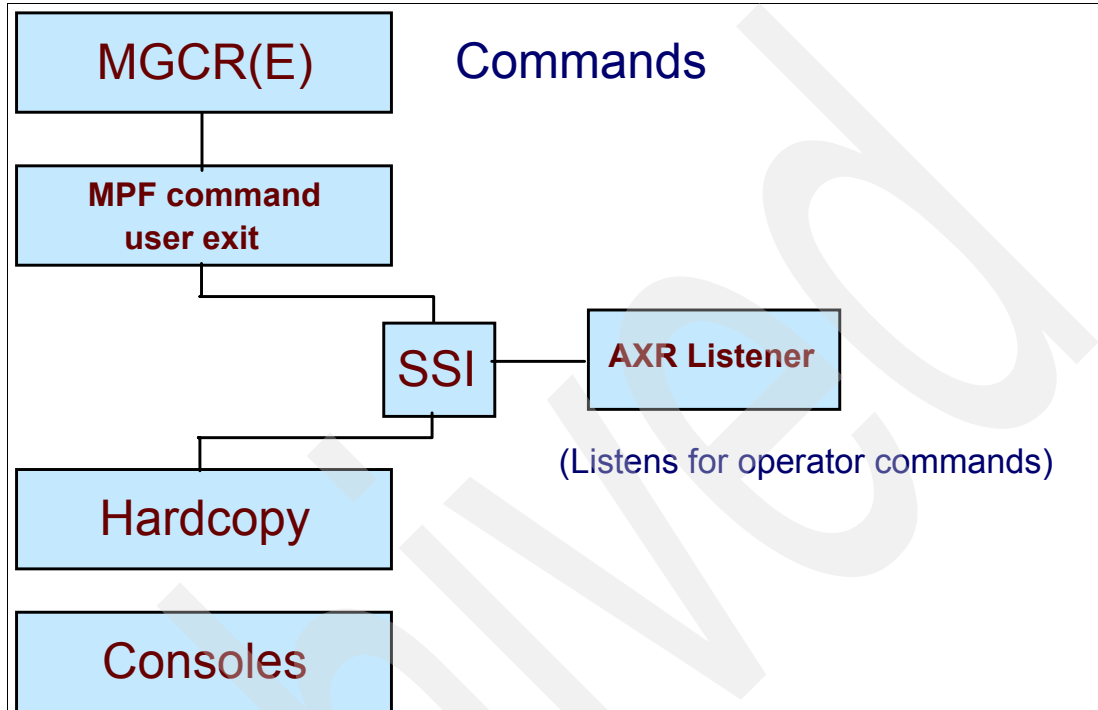


Figure 15-1 AXR listener function that intercepts commands for SYSREXX

## 15.2.2 AXREXX macro service

AXREXX provides a macro interface for System REXX services. Before issuing the AXREXX macro service, the caller does not have to place any information into any general purpose register (GPR) or access register (AR) unless using the input register in register notation for a particular parameter, or using it as a base register.

**Note:** See *z/OS MVS Programming: MVS Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)*, SA22-7609, for a complete description of AXREXX macro parameters.

The AXREXX invoker can limit the amount of time that an exec can run by using the TIMELIMIT/TIMEINT keywords. When the time limit is reached, System REXX invokes HALT interpretation on the REXX environment where the exec is running. If the exec still does not complete after waiting for some time, the task running the exec is detached. Invokers who specify a time limit should realize that time out is an error condition and that for SYNC=YES invokers, the final values of output arguments and variables will not be returned to the AXREXX invoker.

AXREXX supports an interface to **CANCEL** an exec. SYNC=NO AXREXX invokers can obtain the Request Token via the OREQTOKEN parameter for later input to AXREXX **CANCEL** command. Cancel is processed as if the exec timed out.

## Using the AXREXX macro service

The System REXX environment provides a functional package that allows a REXX exec to invoke system commands and to return the results back to the invoker in a variety of ways. When System REXX execs are initiated through an assembler macro interface called AXREXX or through an operator command, there are two different execution environments that are supported:

**TSO=NO** When TSO=NO is specified on the AXREXX invocation, the exec is executed in an MVS host command environment, sharing the address space where it is executing with up to 63 other concurrently running TSO=NO execs. Data set allocation, other than provided by the AXREXX macro, is not supported in the TSO=NO environment.

**TSO=YES** The TSO=YES environment supports all of the host commands that TSO=NO supports, along with some of the host commands supported by TSO/E. If TSO=YES is specified on the AXREXX invocation, the exec will run isolated in a single address space, and can safely allocate data sets without concern of a DDNAME conflict with a concurrently running exec. If the exec were to exit with data sets allocated, System REXX will free the allocations. The TSO environment is established by the dynamic TSO service (IKJTSOEV) and does not support all of the TSO functionality. Running under the MASTER subsystem further restricts what TSO host commands will work.

Applications that perform input/output to data sets other than those specified on the REXXINDSN and REXXOUTDSN AXREXX keywords should use TSO=YES.

As shown in Figure 15-2, the REXX server controls a group of worker subtasks that attach daughter subtasks to process TSO=NO requests. Initially 4 are started, but up to 64 are started as required.

The TSO server controls a group of worker subtasks that start between 1 to 8 address spaces to process TSO=YES requests.

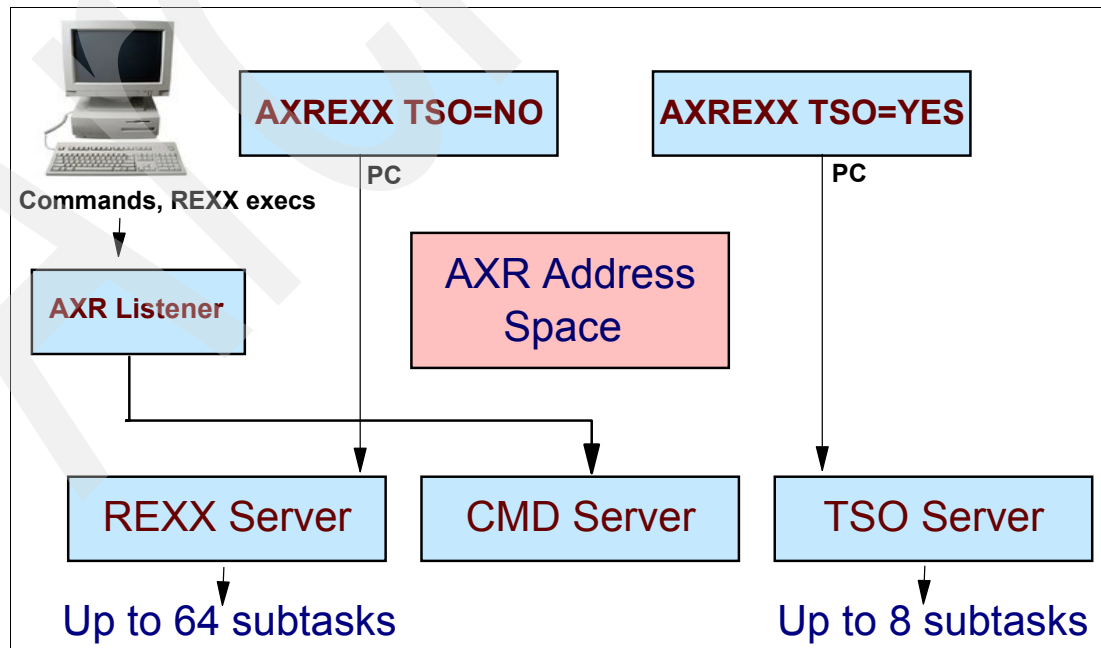


Figure 15-2 System REXX overview of REXX exec processing



**Note:** In both environments the exec runs in problem state, key 8, in an APF authorized address space under the MASTER subsystem; thus any modules that are loaded, linked or attached from the exec must reside in an APF authorized library.

Also in both cases, the REXX exec runs under the WLM enclave of the AXREXX invoker. Neither the TSO=YES nor the TSO=NO environments support UNIX System Services host commands.

## 15.3 Customizing System REXX

The customization necessary to use System REXX involves the AXRxx parmlib member and the user ID to be associated with System REXX and CPF command prefix for issuing console commands.

### AXR00 parmlib member

During AXR address space initialization, the AXRxx parmlib member is read. A sample member is supplied in SYS1.SAMPLIB with member AXR00, as shown in Figure 15-3.

CPF defines a 1 to 8 character command prefix value for System REXX that can be used instead of specifying the **F AXR** command. This prefix may be defined as either SYSTEM or SYSPLEX in scope, where:

- SYSTEM** System scope indicates that the CPF will only be recognized on the system it is defined on.
- SYSPLEX** Sysplex scope indicates that it will be recognized throughout the SYSPLEX, and the command will be routed to the system on which it is defined.

Using the defaults could result in a very unusual prefix. For example, if the SYSCLONE value is 63, then the CPF prefix is REXX63.

```
CPF('REXX&SYSCLONE.',SYSPLEX) /* Defines REXXnn as a sysplex
                               wide cpf value */
AXRUSER(AXRUSER)              /* AXREXX security=axruser results in the
                               exec running in a security environment
                               defined by the userid AXRUSER          */
```

Figure 15-3 SYS1.SAMPLIB member AXR00

Following is parameter setting in the AXR00 member in parmlib to set the CPF character to @ and having a sysplex scope:

```
CPF('@',SYSPLEX) /* DEFINES @AS A SYSPLEX WIDE CPF VALUE */
```

### AXRUSER specification and security

The AXRUSER parameter in the AXRxx parmlib member specifies a 1 to 8 character user ID that is used to define the security environment that an exec initiated using the AXREXX macro when SECURITY=BYAXRUSER is specified. The exec will run with the level of authorization associated with the specified user ID. The installation needs to provide the user ID with SAF access to the resource SYSREXX.<userid>. If this parameter is omitted, then the default is AXRUSER.

The following security considerations are necessary for a SYSREXX environment.

- ▶ AXREXX is an authorized system service, therefore, security controls are essential for:
  - Access to APF library
  - Permissions, standard security administration to determine what can be accessed or run.
- ▶ EXECs by default use the invoker's security environment, but alternatively access can be given as follows:
  - Access authority of a third party
  - A special user ID assigned to AXRUSER
- ▶ EXECs use the invoker's enclave service class, so this can do the following:
  - Prevent CPU priority inversion and excessive resource usage
  - Allow resource usage to be charged back to the enclave service class

The SECURITY and UTOKEN parameters on the AXREXX macro service determine the security environment that the exec runs in. If omitted, the exec will run under the same security environment as its invoker. The security environment determines the data sets that may be accessed and the commands and programs that may be invoked.

- ▶ When SECURITY=BYUTOKEN is specified, the invoker can provide a UTOKEN to define the specific security environment that the exec should run under (see z/OS Security Server RACROUTE Macro Reference).
- ▶ When SECURITY=BYAXRUSER is specified, the exec will run under the security environment associated with the value (user ID) of the AXRUSER parameter specified in the AXR00 parmlib member. This could be useful if the installation wants to invoke AXREXX in an address space that does not have a security environment, such as the MASTER address space. The exec should not invoke any services that alter the security environment of the task running the exec.

In all cases, the REXX exec runs under a WLM enclave of the AXREXX invoker.

## 15.4 Using System REXX

System REXX allows REXX execs to be executed outside of conventional TSO/E and batch environments. The possibilities for exploiting REXX code through the use of System REXX are vast, whether to provide operator assists or to provide an easy way to process files and strings. The System REXX environment provides a function package that allows a REXX exec to invoke the system commands and to return results back to the invoker in a variety of ways. System REXX execs may be initiated through an assembler macro interface called AXREXX, or through an operator command.

### **SYS1.SAXREXEC data set**

The SYS1.SAXREXEC data set contains execs that IBM has provided. These execs should *never* be deleted or modified in any way.

This data set is where user-written execs should be added. These user-written execs should *not* start with the letters A through I, because those letters are reserved for use by IBM.

**Important:** The SYS1.SAXREXEC data set is unique in the system. Therefore, take great care that a regular backup of this SYS1.SAXREXEC data set is taken as follows:

- ▶ User-written execs be easily restored in case of a restore of the resident volume
- ▶ IBM execs be easily restored in case of an human error while editing other members of this PDS

## New functions for writing execs

When executing, the invoked REXX exec has three specific new functions:

**AXRCMD** Issue a console command and return command responses.

AXRCMD is used to issue a system command from within the exec and obtain one or more command responses. The arguments that can be specified are:

- ▶ Command text - the system command to be invoked. This is an optional argument. If it is omitted, no command will be issued, but a response from the last command issuance will be returned if one exists.
- ▶ Msgstem - the stem of a list of variables into which AXRCMD places the command response message text. This is an optional argument. If it is omitted, the command text must be specified. To place the message text into compound variables which allows for indexing, msgstem should end with a period (.) as in "messg." for example.

AXRCMD places each line of the retrieved message into successive variables. For example, if the command response is a 3-line message, messg.1 contains line 1, messg.2 contains line 2 and messg.3 contains line 3. messg.0 will contain the number of lines.

If msgstem does not end with a period, the variable names are appended with consecutive numbers. For example, suppose you specify msgstem as "conmsg" (without a period). If AXRCMD retrieves a message that has two lines of message text, AXRCMD places the text in the variables consmsg1 and consmsg2. The variable consmsg0 contains the number of lines in the message text, which is 2.

- ▶ Time – The amount of time in seconds that AXRCMD should wait for a command response. This is an optional argument. If it is omitted, AXRCMD will not wait before attempting to determine whether a command response was returned. A value of 0 – 21474535 seconds may be specified.

**AXRWTO** Issue a single line message to a console

**AXRMLWTO** Issue a multiline line message to a console

## REXX exec example

Using the new REXX exec functions, Figure 15-4 on page 244 is an example of an exec that issues the **D IPLINFO** command.

```

/* REXX */
MYRESULT = AXRCMD('D IPLINFO',VAR.,5)
GADCON = 'FIRSTLINE'
X = AXRWTO('D IPLINFO')
X = AXRLWTO(SUBSTR(VAR.1,1,50),'GADCON','C')
DO I = 2 TO VAR.0
  X = AXRLWTO(SUBSTR(VAR.I,1,50),'GADCON','D')
END
X = AXRLWTO(,'GADCON','E')
EXIT 0

```

Figure 15-4 Sample REXX exec names IPLINFO in SYS1.SAXREXEC

Figure 15-5 shows the REXX exec IPLINFO issued on a console. The **F AXR,IPLINFO** command gives the same result.

```

@IPLINFO
IEE254I 17.13.09 IPLINFO DISPLAY
SYSTEM IPLED AT 14.34.13 ON 07/09/2007
RELEASE z/OS 01.09.00 LICENSE = z/OS
USED LOADS8 IN SYS0.IPLPARM ON C730
ARCLVL = 2 MTLSHARE = N
IEASYM LIST = XX
IEASYS LIST = (R3,70) (OP)
IODF DEVICE C730
IPL DEVICE D31C VOLUME Z19RC1

```

Figure 15-5 REXX exec @IPLINFO issued on a console

**Note:** When System REXX execs are invoked from a console, the execs are always run in a TSO=YES environment.

### SYSREXX status command

You can use the **F AXR** command to either obtain status about System REXX, or to initiate the execution of a REXX exec. You can also use the prefix defined in the CPF parameter of the AXR00 parmlib member to replace the **F AXR** command. Figure 15-6 on page 245 displays the use of the specified CPY prefix, @, to issue the status command.

```

@SR ST
AXR0200I SYSREXX STATUS DISPLAY 296
SYSTEM REXX STARTED AT 14.35.42 ON 07/09/2007
PARMLIB MEMBERS:      AXR00
CPF: @ (SYSTEM)      AXRUSER: STC
TIMEINT:              30
SUBSYSTEM:           AXR
REQUESTS QUEUED:     0 ACCEPTING NEW WORK
REXX WORKER TASKS:  ACTIVE:  0      TOTAL:  4
                   IDLE:   4      MAX:   64
                   ASYNC:  0      SYNC:   0
                   UNTIMED: 0
TSO SERVER SPACES:  ACTIVE:  0      TOTAL:  0
                   IDLE:   0      MAX:   8
                   ASYNC:  0      SYNC:   0

```

Figure 15-6 *SYSREXX status command*

### **SYSREXX supported commands**

For MVS commands and TSO commands, the following are supported with SYSREXX:

- MVS** LINK, LINKMVS, ATTACH, ATTCHMVS, ATTCHPGM, CPICOMM, LU62, and APPCMVS
- TSO** ALLOCATE (excludes the SYSOUT operand), FREE, ALTLIB, ATTRIB, CALL, DELETE, EXEC, HELP, PROFILE, SMCOPY, and TIME

## **15.5 Usage and migration considerations**

The potential for having SYSREXX available has many possible benefits:

- ▶ It represents an easy way for Web-based servers to run commands or functions and get back pertinent details.
- ▶ It provides a method for system and application components to exploit REXX parsing strengths.
- ▶ It allows you to leverage REXX coding skills, which are extensive and span all operating systems.
- ▶ It allows new health checks to be written quickly.
- ▶ It provides simplified operator assist functions and quick fixes when necessary.
- ▶ It presents exploitative possibilities for customer code, IBM, and ISV components and products.

### **15.5.1 Writing REXX execs**

From a programming point of view, SYSREXX can be invoked by a new AXREXX macro service, and the following list of environments can be used:

- ▶ Caller must be authorized.
- ▶ Two environments of execution:
  - TSO=NO Limited data sets support.

- TSO=YES Full data sets support permitted (no sysout).
- ▶ Two modes of execution
  - SYNC=YES
  - SYNC=NO

When REQUEST=EXECUTE is specified, you can specify an optional parameter that indicates whether the request is synchronous. The default is SYNC=YES.

  - SYNC=YES Indicates the request is synchronous.
  - SYNC=NO Indicates the request is asynchronous.
- ▶ Two operations are supported
  - REQUEST=EXECUTE | CANCEL
- ▶ Arguments and variables are used to pass input to and receive output from the REXX exec.
- ▶ The parameter list is mapped by the AXRZARG macro.
- ▶ Data types supported:
  - Input is converted to strings.
  - Output is returned as signed or unsigned, char, binary, hex.
- ▶ The security environment of requester is used.
- ▶ No STORAGE external function.
- ▶ No dubbing.
- ▶ Time limitations are applied to execs; the default is 30 seconds.
- ▶ Cancellation of the REXX exec using API with OREQTOKEN parameter.
- ▶ REXX execs are read from SYS1.SAXREXEC (cataloged, unique; that is, no concatenation).

**Note:** From a REXX exec perspective, to differentiate whether it is called from a TSO/E environment or from a SYSREXX (either TSO=YES or TSO=NO) environment, you can rely on the contents of AXREQTOKEN. If it is equal to itself, it means that the REXX exec is running in a TSO/E environment. Otherwise, it is running in a SYSREXX environment.

A more detailed example can be found in Appendix B, “System REXX for z/OS” on page 495. The example illustrates how, from within the exec itself, to differentiate when it is running in a TSO=YES or NO environment, and whether it has been invoked from a console or from a program using the AXREXX macro service.

## 15.5.2 Using input and output files

If you are using input and output data sets, they may either be sequential or partitioned. If partitioned, then the REXXINMEMNAME or REXXOUTMEMNAME keywords must be specified. If the output data set does not exist, System REXX creates a sequential or partitioned data set consisting of 3 primary blocks, 3 secondary blocks and 1 directory block (if it is a PDS) where each block is 27920 bytes. The data set is kept when the exec completes, and any excess space is released.

When using the AXREXX macro service, there are many parameters that can be used during processing of the REXX exec, for example:

► Using input data sets

AXREXX allows TSO=NO invokers to pass an input data set via the REXXINDSN parameter and both TSO=NO and TSO=YES invokers to specify an output data set via the REXXOUTDSN parameter. The input data set is used by REXX functions that require input from a user such as PARSE EXTERNAL, or could be read directly via EXECIO, using the DDNAME specified by the REXX variable AXRINDD.

► Using output data sets

If an output data set is specified, any SAY or TRACE output from the exec is directed there. Data may also be written to the output data set via EXECIO using the DDNAME specified by the REXX variable AXROUTDD. Any error messages that the REXX interpreter issues are also directed to the output data set. If no output data set is supplied, then SAY, TRACE, and REXX messages are directed to the console specified by the CONSNM keyword as part of a multi-line WTO AXR0500I.

The AXREXX user should be careful not to flood the system with messages and be especially careful when using REXX Tracing when the output is directed to a console. If CONSNM and REXXOUTDSN are both not specified, the output is lost.

If System REXX detects that the output data set runs out of space, the exec is terminated and a return code of 8 is returned to the AXREXX invoker. If there is no data for the PARSE EXTERNAL command in the input data set, a null string is returned.

### 15.5.3 Other AXREXX parameters

When the request is issued with the SYNC=YES option, the invoker is suspended and the results of the request are provided to the invoker upon resumption. For SYNC=NO, when a failure occurs in attempting to process the exec and System REXX cannot pass a return code back to the AXREXX invoker, message AXR0203I is issued to the console specified on the CONSDATA keyword. If the request is successful, no message is issued.

When command text is specified, AXRCMD invokes the MGCRE macro to issue the command. When the **START** command is invoked, AXRDIAG contains the return code from MGCRE in hexadecimal, followed by the ASID of the new address space (also in hexadecimal), separated by a blank.

### 15.5.4 Arguments and variables within a REXX exec

Within a REXX exec invoked through the AXREXX macro service, 20 arguments and 256 variables are supported. Currently, a maximum supported length is 512 bytes. The total variable storage available for each request is 128 KB.

The REXX exec calling program can, beforehand, allocate storage for variables in one of the following ways:

- In the requester's (or other) address space
- In high virtual storage above the bar
- In dataspaces as ALET qualified addresses are supported

**Note:** The actual location of arguments and variables is totally transparent to the called REXX exec.

## AXRZARG macro service

The AXRZARG macro provides an argument list mapping, and the following special variables are set by AXR:

<b>AXRREQTOKEN</b>	32 Hex	Request Token
<b>AXRINDD</b>	8 Char	If RexxInDsn Specified
<b>AXROUTDD</b>	8 Char	If RexxOutDsn Specified

## Using variables and arguments

The AXREXX macro allows the invoker to specify up to 20 arguments and 256 variables by specifying the REXXARGS or REXXVARS parameter, respectively. To use the REXXARGS and REXXVARS parameters, the AXREXX invoker must create a header section mapped by AXRARGLST followed in contiguous storage by 1 or more AXRARGENTRY sections. The mappings for AXRARGLST and AXRARGENTRY can be found in AXRZARG.

- ▶ AXRARGLST contains the following:
  - AxrArgLstId - set this to either AxrArgLstAcro or AxrVarLstAcro, depending on whether this is for the RexxArgs parameter or the RexxVars parameter.
  - AxrArgLstVer - set to 0 (the current version).
  - AXRARGLstNumber- set to the number of arguments or variables; that is, the number of AXRARGLstEntrys that follow.
  - Other fields must be cleared to zero (0).
- ▶ AXRARGLSTEntry contains the following:
  - AXRARGAddr - set this to the 64-bit address of the buffer containing the argument or variable. If the argument or variable resides below 2 G, use AXRARGAddrLow and make sure AXRARGAddrHigh is zero (0).
  - AxrNameAddr - set this to the 64-bit address of the buffer containing the name of the argument or variable. This field can be set to zero (0) if this is for an input-only argument. If this name resides below 2 G, use AXRNameAddrLow and set AXRNameAddrHigh to zero (0).
  - AXRARGLength - set this to the length of the buffer containing the argument or variable. Note that different argument/variable types have specific requirements regarding lengths.
  - AXRARGAlet - set this to the ALET of the argument/variable. It must be on the DUAL of the task that invokes AXREXX. If the argument/variable resides in the invoker's primary address space, set this to zero (0).
  - AXRARGNameAlet - set this to the ALET of the buffer containing the name of the argument/variable. It must be in the DUAL of the task that invokes AXREXX. If the name resides in the invoker's primary address space, set this to zero (0).
  - AXRArgOutLength - System REXX sets this to the length of data returned to the invoker. Note that this value is in units of bytes for types Signed, Unsigned and Char; it is in units of hex digits (half bytes) for type HexString, and it is in units of bits for type BitString.
  - AxrArgNameLength - set this to the length of the name of the argument. This must contain the actual length of the argument or variable and not include any trailing blanks.
  - AxrArgType - set this to the type of the argument/variable.
  - AXRARGInput - set this if the argument/variable in the REXX exec is to be initialized to a value on entry to the exec.



- AXRArgOutput - set this if you want to retrieve the final value of the argument and variable on exit from the exec for a SYNC=YES request. Note that if the variable is not set by the exec, System REXX will fail the request.
- Other fields must be cleared to zero (0).

## REXX supported data types

Because the only data type in REXX is the character string, System REXX must first convert input arguments or variables into this format. The invoker must specify the data type of the argument or variable in AXRARGTYPE.

The following data types are supported:

- ▶ Unsigned (AXRARGTYPEUNSIGNED) - the input is treated as an unsigned integer value. The length must either be 4 bytes or 8 bytes.
- ▶ Signed (AXRARGTYPESIGNED) - the input is treated as 2s complement signed integer value. The length must either be 4 bytes or 8 bytes.
- ▶ Character (AXRARGTYPECHAR)- the input is treated as a character string. The length can be from 0 to 512 bytes.
- ▶ Hexadecimal (AXRARGTYPEHEXSTRING) - the input is treated as a hexadecimal string. The length is specified in hexadecimal digits (2 per byte) and can be from 0 to 512 hexadecimal digits in length.
- ▶ Bit (AXRARGTYPEBITSTRING) – The input is treated as a bit string. The length is specified in bits (8 per byte), and can be from 0 to 32.

## Error conditions

If AXREXX encounters an error while attempting to control the invoker's input into an REXX argument or variable, System REXX indicates in AXRARGLstEntryInError the number of the argument or variable that caused the error. AXREXX then returns a specific reason code indicating the problem with the argument or variable and abort the request.

If the exec successfully completes (no run time errors), and the AXREXX invocation specifies SYNC=YES, then System REXX will attempt to obtain the final values of any output arguments or variables (that is, those that have indicated AXRARGOUTPUT), convert them into the specified data type and insert their converted values into the AXREXX invoker's buffers specified by AXRARGADDR and AXRARGALET.

If there is any failure with attempting to process a single output argument or variable, System REXX will abort and not attempt to retrieve subsequent arguments or variables. Because output arguments are retrieved prior to output variables, if System REXX fails to process an output argument, no subsequent output arguments are processed and no output variables are processed.

In addition to output arguments and variables, System REXX also returns the return code from the exec in the AXRDIAGEXECRETCODE area in the AXRDIAG (see AXRZARG for the mapping). The return code is returned as a 31-bit signed binary value. If it cannot be converted into such a value, or if the exec does not return a return code, then AXRDIAGNOEXECRETCODE will be set on.

## Variables provided by AXREXX

In addition to any input argument or variables that the AXREXX invoker may provide, System REXX sets the following variables:

- ▶ AXREQTOKEN - contains a 16-byte value which uniquely identifies the AXREXX invocation request.

- ▶ AXRINDD - if the REXXINDSN keyword is specified, this variable will contain the name of the DD used for allocating the input data set; otherwise, it is not set.
- ▶ AXROUTDD - if the REXXOUTDSN keyword is specified, this variable will contain the name of the DD used for allocating the output data set; otherwise, it is not set.



## **z/OS XL C/C++ Metal option**

This chapter describes the performance and usability enhancements provided by z/OS XL C/C++ for the z/OS V1R9 release. Two major areas of support are described: one for developing a C-coded system program (through a new Metal option), and the other for supporting decimal floating-point formats (DFP) assisting in avoiding potential rounding problems. Other usability and performance enhancements are also provided.

## 16.1 Metal option introduction

The z/OS XL C compiler-generated object code relies on the support provided by the Language Environment. In addition to depending on the C run-time library functions that are only available in the Language Environment, the XL C generated object code also depends on the establishment of an overall execution environment, which includes automatic storage. This dependency on the Language Environment prohibits you from using the XL C compiler to generate code to run in an environment where the Language Environment does not exist.

Prior to z/OS V1R9, all z/OS XL C compiler-generated code required Language Environment. In addition to depending on the C runtime library functions that are available only with Language Environment, the generated code depended on the establishment of an overall execution context, including the heap storage and dynamic storage areas. These dependencies prohibit you from using the XL C compiler to generate code that runs in an environment where Language Environment did not exist.

The Metal C Runtime Library is a set of LPA-resident C functions that can be called from a C program created using the z/OS XL C compiler Metal option. This is a new base element in z/OS V1R9.

### 16.1.1 XL C Metal compiler option

The XL C Metal compiler option, introduced in z/OS V1R9, generates code that does not have access to the Language Environment support at run time. Instead, the Metal option provides C language extensions that allow you to specify assembly statements that call system services directly. Using these language extensions, you can provide almost any assembly macro, and your own function prologs and epilogs, to be embedded in the generated HLASM source file. When you understand how the Metal-generated code uses MVS linkage conventions to interact with HLASM code, you can use this capability to write freestanding programs.

Figure 16-1 on page 253 lists the overall enhancements for IBM z/OS XL C/C++.

The Metal-generated object code does not depend on any run-time environment. Any system services that the program needs can be obtained directly by using the system macros supplied by the operating system.

XLC Arch ( )	Hardware facility	Hardw. arch.	machine models	XLC min level	XLC option	Note
	AR-mode	31/64		1.9	METAL	no LE Envrt
	decimal floating point		z9	1.8	DFP	LANGLVL (EXTENDED)
7	extended-immediate facility		z9			x- xlation 3
6	long/displacement facility		z990 z890			x - xlation 2
5	64-bit mode	ESAME	z900 z800	1.2		default for TARGET(z/OS 7) and above
4	long long operations	ESA/390	z900 z800			
3	IEEE (binary) floating-pt		G5-G6			
2	Branch Relative and Halfword Immediate		(G2-G4)			
1	Logical String Assist		G1 9021			
0	produced code is executable on all models					

Figure 16-1 Overall enhancements for IBM z/OS XL C/C++

### C/C++ support for decimal floating-point formats

z/OS XL C/C++ supports, in release 8 and 9, the decimal floating-point (DFP) formats in addition to the current hex and binary floating-point formats. This support is available on the IBM System z9 BC and IBM System z9 EC models, and is activated at the XL compiler in C/C++ via a new DFP option when LANGLVL is set to EXTENDED. The decimal formats are specified by the revised IEEE 754 floating point standard.

This support assists with avoiding potential rounding problems, which can result from using binary or hexadecimal floating-point types to handle decimal calculations. z/OS XL C/C++ provides the following:

- ▶ Decimal type specifiers through the `_Decimal32`, `_Decimal64`, and `_Decimal128` reserved keywords
- ▶ Decimal literal support
- ▶ Conversion between decimal types and the other floating-point types

## 16.2 XL C Metal option

The XL C compiler generated code needs Language Environment to execute. However, a need exists to run C programs where Language Environment is not available or is undesirable, such as with the long-awaited support for AR mode programming.

Therefore, the XL C compiler provides a new option called Metal, reflecting the idea of a programming environment closed to the bare Metal (or silicon) of the chip. This new option allows the opportunity to do the following:

- ▶ Enable code generation with no Language Environment dependencies
- ▶ Provide support for programmer supplied prolog/epilog code

- ▶ Provide support for programmer-embedded HLASM statements within the C code
- ▶ Provide support for AR mode programming from the C language

**Note:** The Metal option is only available for programs written in strict C language. There is no support whatsoever for C++ programs.

## 16.2.1 Metal option overview

A new mode of code generation is added to the XL C compiler. The Metal option serves as the switch to enable this new mode of code generation so that the code generated follows the standard MVS linkage conventions and does not have Language Environment dependencies.

In the new mode of code generation, you can specify:

- ▶ The #pragma prolog/epilog directives that can be used to supply a programmer's own prolog and epilog code
- ▶ The \_\_asm syntax that can be used to supply embedded HLASM statements
- ▶ The new language constructs and facilities to support accessing data stored in data spaces
- ▶ The subset of C library functions available for this mode

With this new mode of code generation, it is therefore possible to use C language to:

- ▶ Write installation exits
- ▶ Develop programs that runs without Language Environment assistance
- ▶ Have a much simpler way of writing AR mode programs

By definition, a C function needs a stack space to store function scope variables and temporary storage for compiler-generated code. Normally, Language Environment supplies the stack space. When the generated code has no Language Environment at hand, the stack space may need to be supplied by the programmer.

Without Language Environment, C library functions are unavailable. Thus, there needs to be a way for the C function to enlist system services directly. Therefore, the code has to be compatible with the linkage expected by corresponding MVS system components. For more information about this topic, refer to *z/OS Metal C Programming Guide and Reference*, SA23-2225.

## 16.2.2 Using the Metal option

The term "Metal" can denote "raw" and "fundamental". Thus, the code generated by this option should be the fundamental code sequence that can run with minimal environmental dependencies, as follows:

- ▶ The linkage convention mostly follows those described in *z/OS MVS Programming: Assembler Services Guide*, SA22-7627.
- ▶ With the Metal option, other capabilities are enabled such as supplying a user's own prolog and epilog code, embedding a user's own HLASM statements in the C source code, and AR mode programming.
- ▶ Some other compiler options become meaningless (such as XPLINK, DLL, RENT, and so on).

- ▶ The application build process now requires an additional step to invoke the assembler to assemble the compiler-generated code; see Figure 16-2. An equivalent JCL procedure can be found in Appendix A, “Metal option of XL C compiler” on page 491.

```
xlc -S -qMetal foo.c produces foo.s
as -c foo.s produces foo.o
ld -o foo -e MAIN foo.o produces the executable foo
```

Figure 16-2 New sequence in application build

- ▶ The C-generated programs can now be mixed with programs written in HLASM.
- ▶ AMODE 64 code is also supported.
- ▶ z/Architecture hardware is required.

### GENASM option

To allow programmer-injected HLASM statements, the compiler needs to produce code in HLASM source code. The GENASM option can be used to name the output HLASM source file.

The GENASM option also enables the compiler to process the `__asm` statements. The `-S` flag on the `xlc` command is equivalent to specifying the GENASM option. Currently, the GENASM option can only be used with the Metal option.

### Programmer-supplied prolog and epilog code

The compiler generates default prolog and epilog code which operates on a 1 M stack space. It may be necessary for the programmer to supply prolog and epilog code to set up whatever the environment is.

There are several ways to supply customized prolog and epilog code:

- ▶ Using the `#pragma prolog/epilog` directives to apply the prolog/epilog to a single function
- ▶ Using the `PROLOG` and `EPILOG®` options to apply the same prolog and epilog to all extern functions in the source file

Global set symbols are used to communicate compiler information to the user code; the intended use of this capability is for the programmer to specify a macro that contains the prolog/epilog code.

### Programmer-embedded HLASM statements

An example of a `printf` like `__asm` statement for embedding user HLASM statements using the following syntax is shown here.

```
__asm ( code_format_string : output : input : clobbers)
```

Where:

- ▶ `code_format_string` is a string literal similar to a `printf` format specifier.
- ▶ `output` is a comma-separated list of output operands; the list is optional.
- ▶ `input` is a comma separated list of input operands; the list is optional.
- ▶ `clobbers` is a comma-separated list of clobber registers; the list is optional.

The colons separating output, input, and clobbers are required because these components are specified by position. Any or all of these components can be omitted. Trailing colons as a

result of omitting one or more components can be omitted. Input and output have the following general syntax.

```
constraint (C_expression) , constraint (C_expression) , ...
```

Each constraint (C\_expression) pair represents one operand. Zero (0), one, or more can be specified, separated by commas.

Keep in mind that using standard hw\_builtin functions is preferred, rather than using \_\_asm statements. Overall, the programmer is fully responsible for the correctness of the supplied HLASM statements. This \_\_asm function can be called from a C program, using the Metal option, in the following way.

```
int main() {
    struct WTO_PARM {
        unsigned short len;
        unsigned short code;
        char text[80];} wto_buff = { 4+11, 0, "hello world" };
    __asm( " WTO MF=(E,(%0)) " : : "r"(&wto_buff));

    return 0;
}
```

### 16.2.3 Linkage conventions

The standard register convention used is as follows:

- ▶ The 72-byte standard save area is used for AMODE 31 code.
- ▶ The 144-byte F4SA save area is used for AMODE 64 code.
- ▶ The 216-byte F7SA save area is used when the function has calls to AR mode functions.
- ▶ A Next Available Byte (NAB) convention is adopted to supply stack space to the called functions. To use NAB, a sizable stack frame has to be established for the whole program calling chain; meanwhile, there is no provision for detecting the stack floor condition.

#### Prolog and epilog code

The compiler generates default prolog and epilog code which operates on a 1 M stack space. It may be necessary for the programmer to supply a specific prolog or epilog code to set up whatever environment is necessary.

There are two ways to supply customized prolog and epilog code:

- ▶ Using the #pragma prolog and epilog directives to apply the prolog and epilog to a single function
- ▶ Using the PROLOG and EPILOG options to apply the same prolog and epilog to all extern functions in the source file.

### 16.2.4 AR-mode and the Metal option

The XL C compiler provides AR-mode programming support under the Metal option. An AR-mode function can access data stored in data spaces by using the hardware access registers.

For more information about AR-mode, see *z/OS MVS Programming: Assembler Services Guide*, SA22-7605. A non-AR-mode function is said to be in primary mode.



A C language construct is provided for:

- ▶ Marking a function to be an AR mode function by the `__attribute__((armode))` attribute
- ▶ Qualifying a pointer to be a far pointer by the `__far` qualifier.

There is also an option `ARMODE` for marking every function in the source file to be AR-mode functions. Built-in functions are provided for manipulating far pointers. The constructors are as follows:

- ▶ `void * __far __set_far_ALET(unsigned int alet, void * offset);`
- ▶ `void * __far __set_far_ALET(unsigned int alet, void * __far offset);`
- ▶ `void * __far __set_far_offset(void * __far alet, void * offset);`

The extractors are as follows:

- ▶ `unsigned int __get_far_ALET(void * __far p);`
- ▶ `void * __get_far_offset(void * __far p);`

Standard string and memory functions are provided as “far-versions” for processing data stored in data spaces.

## 16.2.5 Metal C runtime library

Although the compiler generates default prolog and epilog code that allows the Metal C code to run, you might be required to supply your own prolog and epilog code to satisfy runtime environment requirements. Metal C avoids excessive acquisition and release operations by providing a mechanism that allows a called function to rely on preallocated stack space. This mechanism is the next available byte (NAB). All Metal C runtime library functions, as well as functions with a default prolog code, use it and expect the NAB address to be set by the calling function.

A new runtime library provides useful utility functions for users of the Metal C compiler such as:

- ▶ Utility functions for manipulating data
- ▶ Memory management
- ▶ Does not support floating point or I/O
- ▶ Supports AMODE 31 and AMODE 64
- ▶ Requires Primary ASC mode (except for “far-versions”, all data has to be in primary address space)

The Metal C runtime library has LPA-resident functions which are made available during the IPL and are called through *system vectors*. Corresponding standard header files are provided for these functions which are completely independent of Language Environment.

A starter set of standard C functions is provided, such as:

- ▶ Data manipulation, conversion
- ▶ Character classification
- ▶ Memory management

For a complete list of functions, refer to *z/OS Metal C Programming Guide and Reference*, SA23-2225.

## 16.3 Decimal floating point

Decimal arithmetic is the norm in human calculations, and human-centric applications must use a decimal floating-point arithmetic to achieve the same results.

Initial benchmarks indicate that some applications spend 50% to 90% of their time in decimal processing, because software decimal arithmetic suffers a 100× to 1000× performance penalty over hardware. The need for decimal floating-point in hardware is there.

Existing designs, however, either fail to conform to modern standards or are incompatible with the established rules of decimal arithmetic. This chapter introduces a new approach available in the IBM System z9 models and provided by z/OS V1R9, consisting of decimal floating-points—which not only provides the strict results necessary for commercial applications, but also meets the constraints and requirements of the IEEE 854 standard that is being finalized.

The hardware implementation of this arithmetic is expected to significantly accelerate a wide variety of applications.

### 16.3.1 The need for decimal arithmetic

Despite the widespread use of binary arithmetic, decimal computation remains essential for many applications. Not only is it required whenever numbers are presented for human inspection, but it is also often a necessity when fractions are involved.

Decimal fractions (rational numbers whose denominator is a power of ten) are pervasive in human endeavours, yet most cannot be represented by binary fractions; the value 0.1, for example, requires an infinitely recurring binary number. If a binary approximation is used instead of an exact decimal fraction, results can be incorrect even if subsequent arithmetic is exact.

For example, consider a calculation involving a 5% sales tax on an item (such as a \$0.70 telephone call), rounded to the nearest cent. Using double-precision binary floating-point, the result of multiplying 0.70 by 1.05 is a little under 0.7349999999999999, whereas a calculation using decimal fractions would yield exactly 0.735. The latter would be rounded up to \$0.74, but using the binary fraction the result returned would be the incorrect \$0.73.

For this reason, financial calculations (or, indeed, any calculations where the results achieved are required to match those which might be calculated by hand), are carried out using decimal arithmetic.

Further, numbers in commercial databases are predominately decimal. Commercial databases contain identifiably numeric data, and of these the majority is decimal (such as the SQL NUMERIC). A large number of those that are non-decimal are integer types which could have been stored as decimals.

### 16.3.2 Extended precision floating-point numbers

Since introduction, z/Architecture predecessors have introduced different formats of floating-points which can be depicted as shown in Figure 16-3 on page 259.

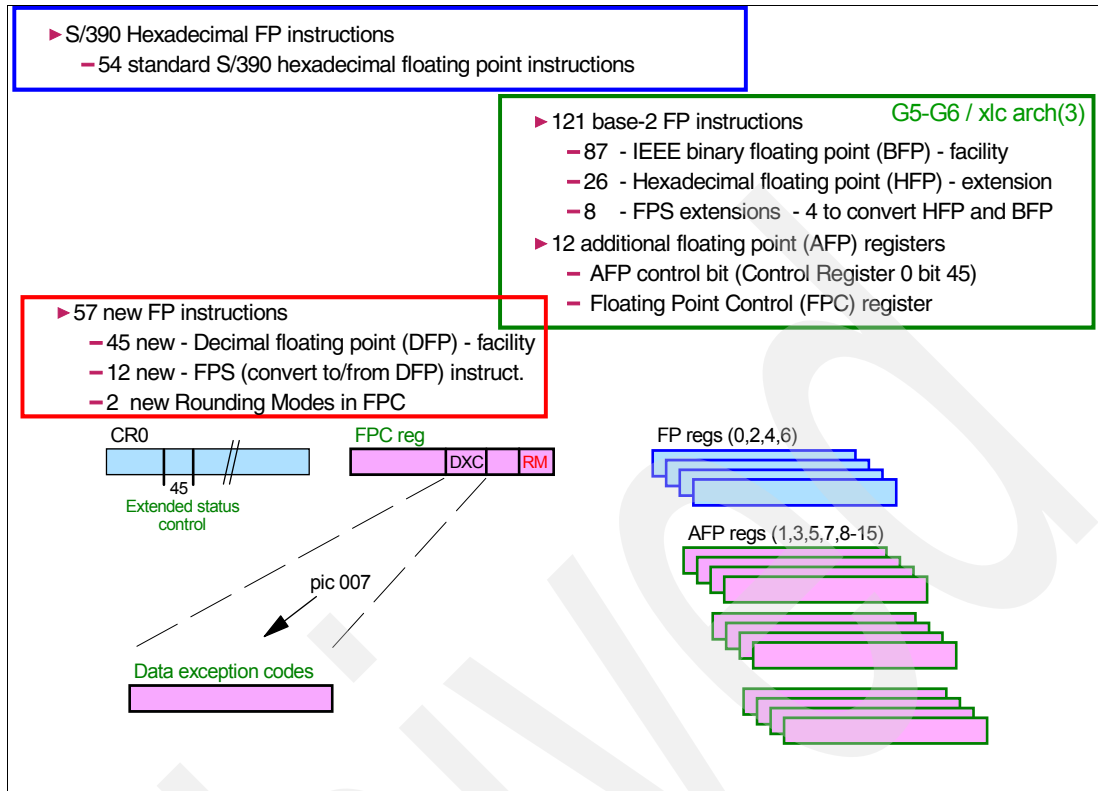


Figure 16-3 z/Architecture floating points

### 16.3.3 New floating-point data types

z/OS XL C/C++ provides three z/Architecture floating-point number data types:

- Single precision (32 bits), declared as float
- Double precision (64 bits), declared as double
- Extended precision (128 bits), declared as long double

Extended precision floating-point numbers give greater accuracy to mathematical calculations.

z/OS XL C/C++ also supports IEEE 754 floating-point representation (base 2 or binary floating-point formats). By default, float, double, and long double values are represented in z/Architecture floating-point formats (base-16 floating-point formats).

However, the IEEE 754 floating-point representation is used if you specify the `FLOAT(IEEE)` compiler option.

As of z/OS V1R9, XL C/C++ also supports IEEE 754 decimal floating-point representation (base-10 floating-point formats), with the types `_Decimal32`, `_Decimal64`, and `_Decimal128`, if the `DFP` compiler option is specified.

### 16.3.4 Decimal arithmetic context

Decimal arithmetic is implemented as a set of specific operation codes handling DFP number operands stored under a defined encoding format. The whole is handled as part of the

common I-stream of the application, but within a context to which two characteristics have been added: commercial rounding and precision, as explained in the following sections.

### Commercial rounding

The extra rounding mode is known as *round-half-up*, which is a requirement for many financial calculations (especially for tax purposes and in Europe). In this mode, if the digits discarded during rounding represent greater than or equal to half (0.5) of the value of a one in the next left position, then the result should be rounded up. Otherwise, the discarded digits are ignored.

This is in contrast to round-half-even, the default IEEE 854 rounding mode, where if the discarded digits are exactly half of the next digit, then the least significant digit of the result will be even.

z/Architecture implementations offer two further rounding modes:

- ▶ Round-half-down (where a 0.5 case is rounded down)
- ▶ Round-up (round away from zero)

The rounding modes in IEEE 854, together with these three modes, are the same set as those available in Java.

### Precision

The working precision setting in the context is a positive integer which sets the maximum number of significant digits that can result from an arithmetic operation. It can be set to any value up to the maximum length of the coefficient, and lets the programmer choose the appropriate working precision.

In the case of software (which may well support unlimited precision), this lets the programmer set the precision and hence limit computation costs. For example, if a daily interest rate multiplier  $R$  is 1.000171 (0.0171%, or roughly 6.4% per annum), then the exact calculation of the yearly rate in a non-leap year is  $R^{*365}$ .

Calculating this to give an exact result requires 2191 digits, but a much shorter result that is correct to within one unit in the last place (ulp) will almost always be sufficient and could be calculated much faster.

In the case of hardware, precision control has little effect on performance, but it does allow the hardware to be used for calculations of a different precision from the available “natural” register size. For example, one proposal for a concrete representation suggests a maximum coefficient length of 33 digits; this would be unsuitable for implementing the new COBOL standard (which specifies 32-digit intermediate results) if precision control in some form were not available. Note that to conform to IEEE 854 §3.1 the working precision should not be set to less than 6.

## 16.3.5 XL C/C++ support for decimal floating point data types

Decimal floating point data types are provided by XL C/C++ of z/OS V1R9 under the form of `_Decimal32`, `_Decimal64` and `_Decimal128` for single, double and quad precision respectively. New suffixes for Decimal Floating Point literal values are `df`, `dd`, `dl`, or `DF`, `DD`, and `DL`.

Hardware decimal floating point instructions are generated to carry out decimal floating point operations. A set of built-in functions is provided to allow specific hardware decimal floating

point instructions to be generated. Debug support for the new Decimal Floating Point data types is also provided.

Overall, C/C++ applications can utilize the new decimal floating point data types, which allows users to avoid the rounding problems associated with the typical binary representation of fractional data.

### 16.3.6 XL C/C++ run-time library

The C/C++ run-time library (RTL) is enhanced to provide:

- ▶ New functions and macros for getting and setting the rounding mode for decimal floating point operations.
- ▶ New macros for defining decimal floating point limits and evaluation formats.
- ▶ Updated floating point environment functions for manipulating the decimal floating point environment.
- ▶ New and updated decimal floating point math functions and macros.
- ▶ New functions for converting character strings, and wide character strings, to decimal floating point types.
- ▶ New functions for casting floating point numbers between binary, hexadecimal, and decimal floating point.
- ▶ New optional prefixes for indicating the size of decimal floating point arguments for printf() and scanf() families of functions, with the following warning:

### 16.3.7 UNIX System Services dbx debugger

Debugger dbx, part of UNIX System Services, supports decimal floating point data types so that customers can debug decimal floating point data types and register representations in their applications. Expression@ handling and the assignment and display of decimal floating point data types in program data and registers are provided by dbx.

Support for debugging of `_Decimal32`, `_Decimal64`, `_Decimal128` data types and new register representations, such as new register symbols, as follows:

```
$frdX with X = 0..15
print $frd5
assign $frd8=2.2
```

Value formatting is handled the same way as other primary data types

- ▶ “print” subcommand (formats value)
  - print mydec64
  - 2.2003
- ▶ “whatis” subcommand (formats definition)
  - whatis mydec64
  - `_Decimal64 mydec64;`
- ▶ dbx expressions such as ‘mydec64+2.3’ are promoted internally to long double binary, same as other floating point data.
- ▶ Evaluated DFP register precision is set by `$fl_precision` (4, 8 or 16); the default is 8.

## 16.4 dbx support of WebSphere remote debuggers

dbx supports the IBM WebSphere Developer for System z (WDz), and the IBM WebSphere Developer Debugger for System z (WDDz), both at V7.0.

Both graphical user interfaces run on user workstations under Windows.

- ▶ The WDz or WDDz remote debuggers user interface runs on the user's workstation (under Windows) and communicates with the dbx engine through a TCP/IP socket running on z/OS that is debugging the z/OS application or dump.

The program being debugged or dump, source files and debug information resides on the z/OS server running the dbx engine. The socket shell is the dbx shell that will communicate with a user interface over TCP/IP sockets, as depicted in Figure 16-4 on page 262, where:

- To the left of the “dbx Engine” block is the current Command Line Shell that gives the user a line/page interface to their terminal.
  - To the right of the “dbx Engine” block is the new socket shell that communicates over a TCP/IP link to the Remote Debugger UI (WDz or WDDz) that is running on their workstation.
  - The dbx engine at the core remains the same for both shells.
- ▶ Start up WDz/WDDz UI on a specific port, then use the `-p` option to tell dbx the machine name or IP address and the port to connect to.
    - Any further user interactions to the dbx engine will be done from the user interface.
    - If no port is specified, the default port of 8001 will be used.

Figure 16-4 on page 262 illustrates the dbx remote debuggers support.

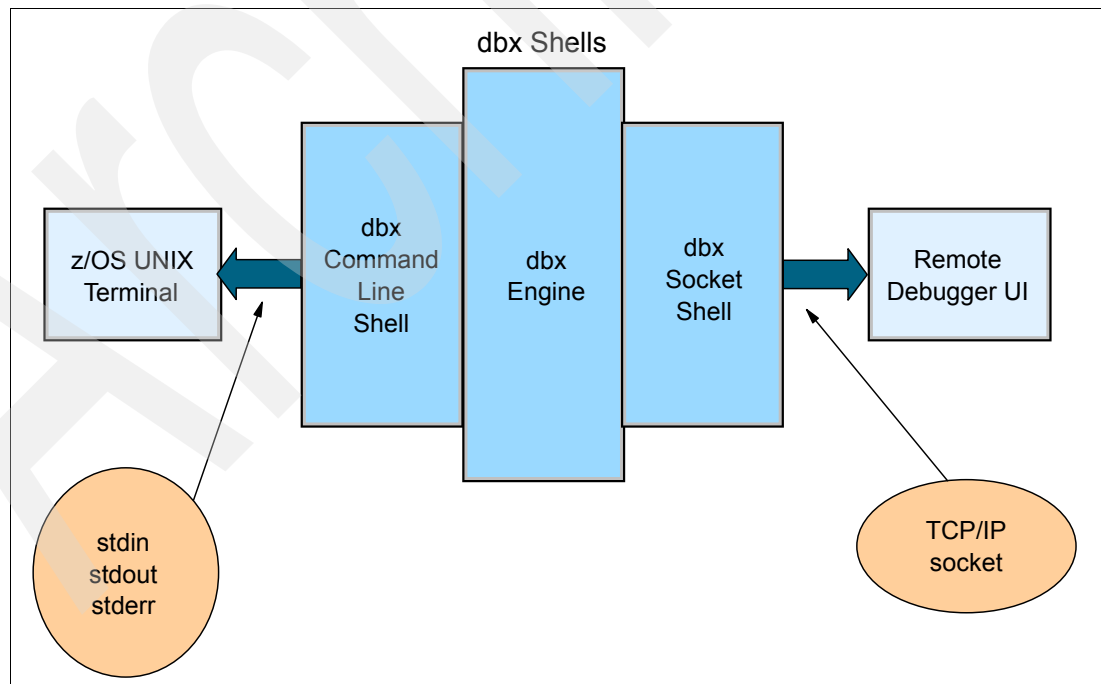


Figure 16-4 dbx remote debuggers support

## dbx debugging example

Figure 16-5 on page 263 and Figure 16-6 on page 264 are examples of using WDDz as the user interface to the dbx engine.

- ▶ A typical debugging session is in progress for a source level view of a program.
- ▶ The Debug window shows one thread, the program being debugged, and that main() called function f1(), then f2(), then f3(), and that dbx is the target engine on a z/OS system.
- ▶ The Source window (with title tsimple.c) shows the entry breakpoint in f3(), which is the current source line and a line breakpoint at 22.
- ▶ The Breakpoint window shows that an entry breakpoint and a line breakpoint have been set.
- ▶ The Modules window show there is currently one .c file in this simple program with four functions.
- ▶ The Variable window shows there are three local variables in f3(). Variables that have children (struct/union/class/array) will have a plus [+] sign next to the name to indicate it can be expanded.
- ▶ The Monitors window shows we set various expression monitors with their current values.
- ▶ Right-clicking most items will bring up an action menu. For example, right-clicking a line in the Source window will allow line breakpoint to be set.
- ▶ Right-clicking a variable value will allow the value to be changed.
- ▶ Various execution options are supported such as step into, step over, step return and resume.

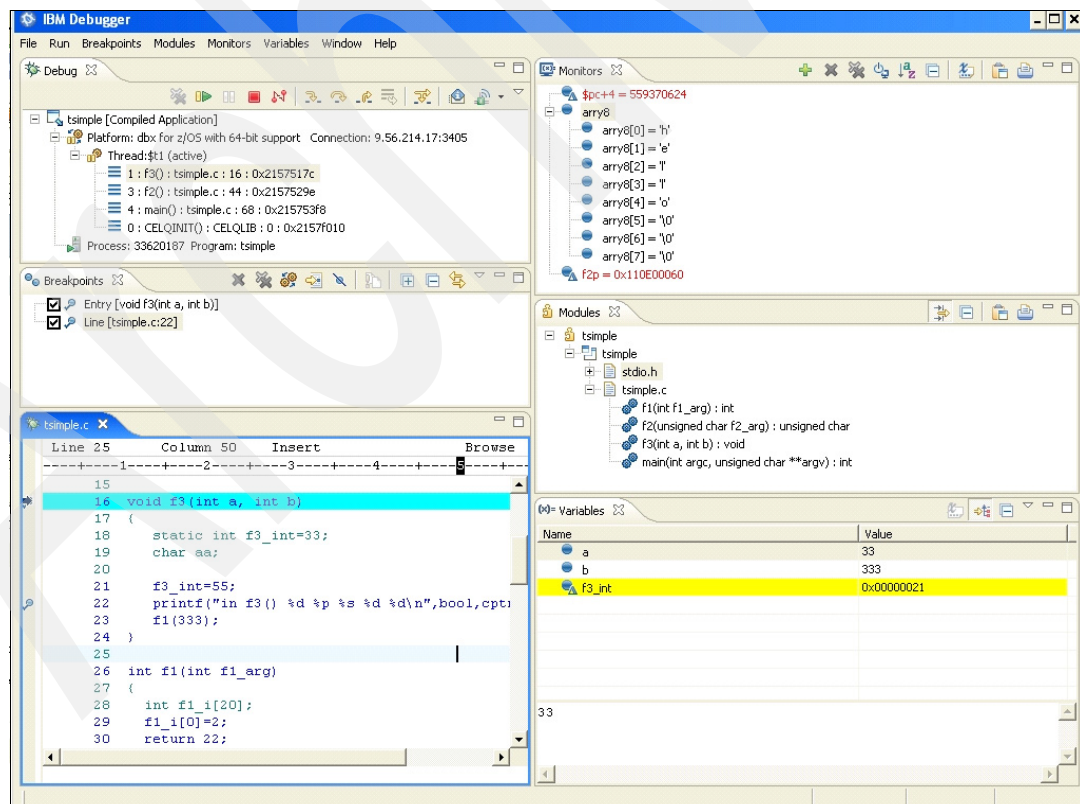


Figure 16-5 dbx remote debugger example

## dbx remote debugging example

Figure 16-6 on page 264 shows a typical debugging session in progress for a machine level view of a program.

- ▶ The Debug window shows one thread, the program being debugged, and that main() called function f1(), then f2(), then f3(), and that dbx is the target engine on a z/OS system.
- ▶ The Source window (with title tsimple\_f3\_320.dsm) shows the disassembly view with an address breakpoint.
- ▶ The Breakpoint window shows an address breakpoint.
- ▶ The Memory window shows a hex and EBCDIC representation of the storage.
- ▶ The Registers window shows the PSW group and the GPR register group expanded. Nine other register groups are for the floating point registers to show HFP, BFP, and DFP representations of the floating point registers in float, double, and long double sizes.
- ▶ Right-clicking most items will bring up an action menu. For example, right-clicking a line in the Disassembly window will allow an address breakpoint to be set.

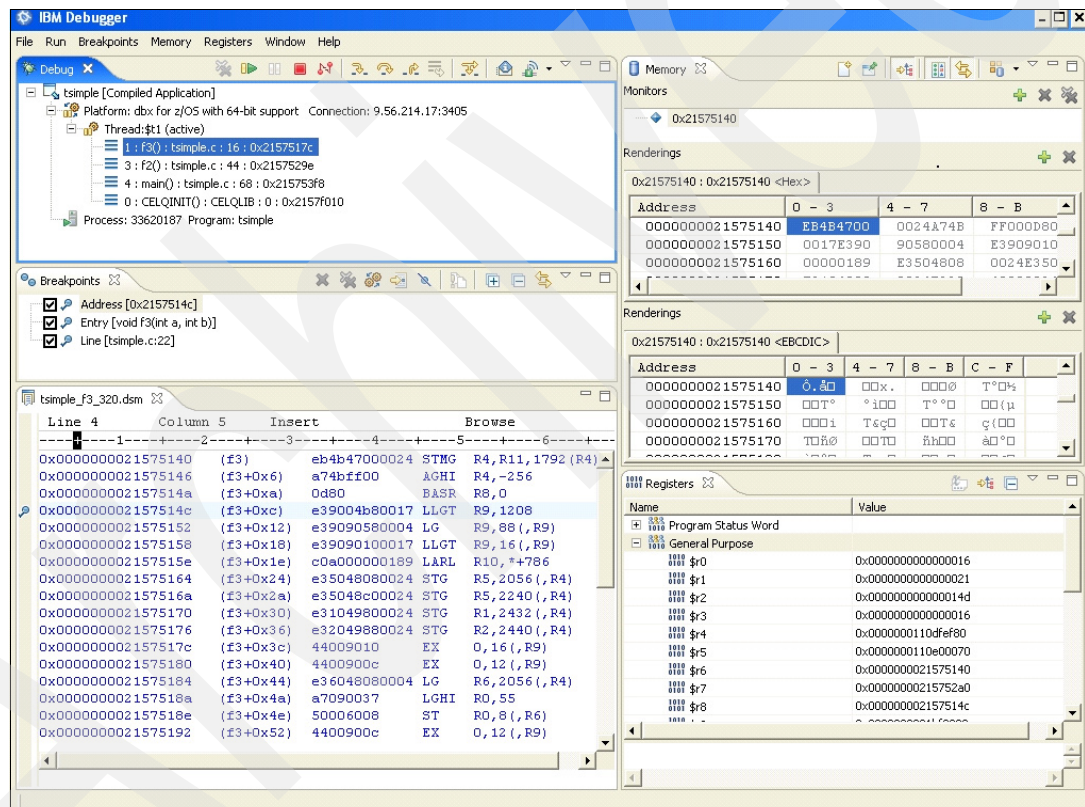


Figure 16-6 Another example of dbx remote debugging

## dbx debugging example

Figure 16-7 on page 265 shows a typical debugging session in progress for a source level view of a multi-threaded program.

- ▶ The Debug window shows three threads, the program being debugged and the selected stack context in the second thread.
- ▶ The Source window shows the source location for the current stack frame and the current thread.
- ▶ The Debug Console shows two dbx commands were entered and the output.



- ▶ The properties shows the properties for the selected item (in this case, the stack information for the selected stack frame).

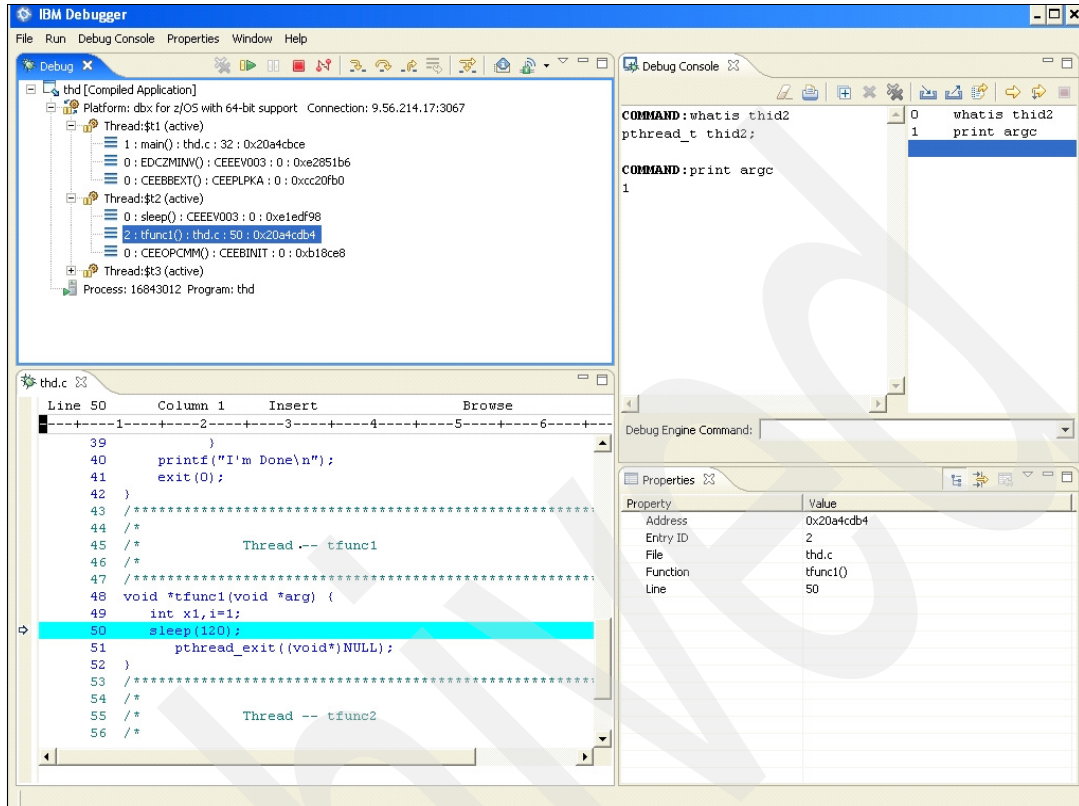


Figure 16-7 dbx remote debugging of a multi-threaded program

### dbx debugging example

Figure 16-8 on page 266 is a typical debugging session in progress for a source level view of a multi-process program.

- ▶ The program forked and the user selected “follow both” in a GUI displayed pop-up.
- ▶ The fork pop-up also allows “follow parent” and “follow child” selections.
- ▶ The Debug window shows two processes with a focus on a stack frame in the child.
- ▶ On a fork follow both, dbx forks off a new dbx and does a “dbx -A” to the child, connecting back to the UI on the same port to allow debugging of both the parent process and the child process.
- ▶ On a fork follow parent, dbx detaches the child and allows the child to run without any further dbx interaction and overhead. The parent can continue to be debugged.
- ▶ On a fork follow child, dbx detaches the parent and allows the parent to run without any further dbx interaction and overhead. The child can continue to be debugged.

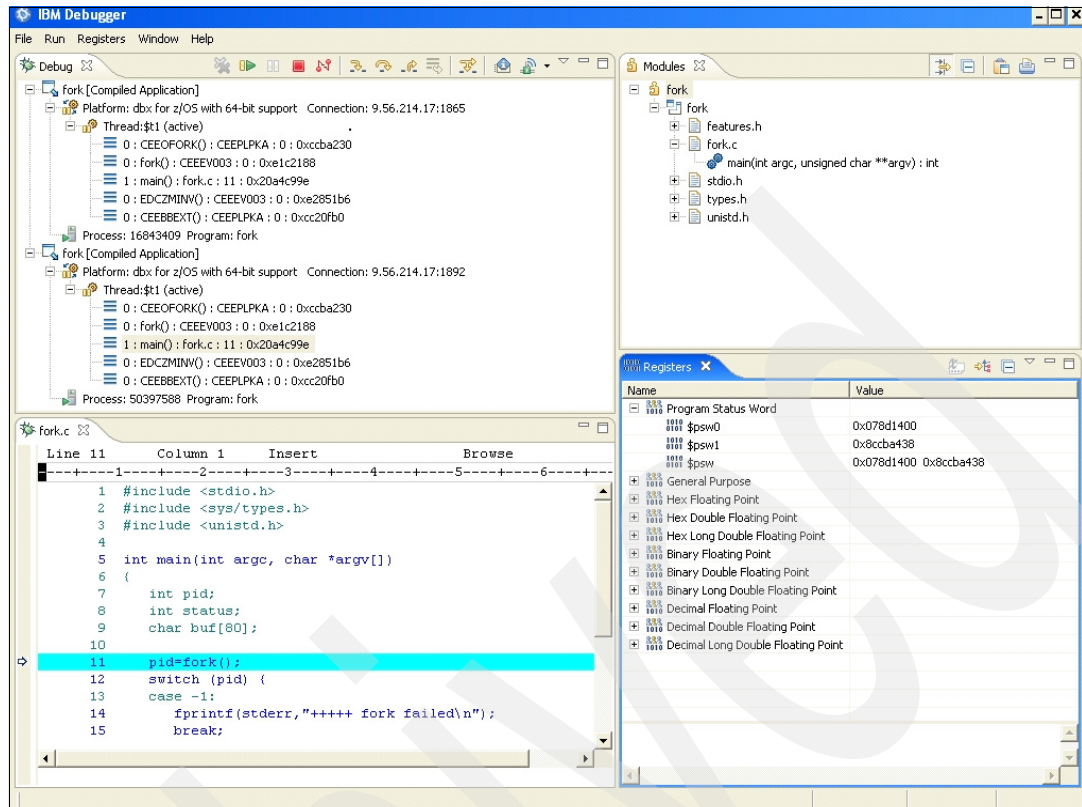


Figure 16-8 dbx remote debugging of a multi-process program

## 16.5 Specialized hardware instructions support

XL C/C++ in z/OS V1R9 provides access to specialized hardware instructions simplifying software logic. In this way, the XL C/C++ compilers pursue the goal of more optimized code by providing:

- ▶ A new set of hardware instruction built-in functions
- ▶ Fine tuning of instruction selection and sequence

The net benefit is better performing C/C++ applications.

A new set of built-in functions, as explained here:

Instructions from the extended-translation facilities (1, 2, and 3) which may be available on a model implementing z/Architecture.

- ▶ Extended-translation facility 1, introduced in ESA/390 (ARCH(5)=z900), is standard in z/Architecture and provides the instructions:
  - CONVERT UNICODE TO UTF-8
  - CONVERT UTF-8 TO UNICODE
  - TRANSLATE EXTENDED
- ▶ Extended-translation facility 2 (ARCH(6)=z990) performs operations on double-byte, ASCII, and decimal data. The double-byte data may be Unicode(c) data that uses the binary codes of the Unicode Worldwide Character Standard and enables the use of

characters of most of the world's written languages. The facility provides the following instructions:

- COMPARE LOGICAL LONG UNICODE
- MOVE LONG UNICODE
- PACK ASCII
- PACK UNICODE
- TEST DECIMAL
- UNPACK ASCII
- UNPACK UNICODE
- ▶ Introduced by ETF2-Enhancement Facility (ARCH(7)):
  - TRANSLATE ONE TO ONE
  - TRANSLATE ONE TO TWO
  - TRANSLATE TWO TO ONE
  - TRANSLATE TWO TO TWO
- ▶ For cases where either or both facility 1 and facility 2 are not installed on the machine, both facilities are simulated by the MVS CSRUNIC macro instruction, which is provided in OS/390 Release 10 and z/OS.
- ▶ Extended-translation facility 3 ((ARCH(7)=z9) may be available on a model implementing z/Architecture. The facility performs operations on Unicode and Unicode-transformation-format (UTF) characters. It also includes a right-to-left TRANSLATE AND TEST operation. The facility provides the following instructions:
  - CONVERT UTF-16 TO UTF-32
  - CONVERT UTF-32 TO UTF-16
  - CONVERT UTF-32 TO UTF-8
  - CONVERT UTF-8 TO UTF-32
  - SEARCH STRING UNICODE
  - TRANSLATE AND TEST REVERSED
- ▶ Convert-to-Binary (CVB and CVBG) and Convert-to-Decimal (CVD and CVDG).
- ▶ Compare-Logical-Long-Extended (CLCLE) and Move-Long-Extended (MVCLE).
- ▶ Zero-and-Add (ZAP).

New application capabilities are provided in dealing with UNICODE data and other special purpose operations. Instruction selection and scheduling have been fine tuned to avoid Address Generation Interlock (AGI) delays.

### 16.5.1 Available new built-in functions

The corresponding new built-in functions are available as follows:

- ▶ ARCH(0) option: CVB, CVD, ZAP
- ▶ ARCH(2) option: CLCLE, MVCLE
- ▶ ARCH(5) option: CVBG, CVDG, TRE
- ▶ ARCH(6) option: CLCLU, MVCLU, PKA, UNPKA, PKU, UNPKU, TP

- ▶ ARCH(7) option:
  - TROO, TROT, TRTO, TRTT
  - CU41, CU42, CU12, CU14, CU21, CU24
  - SRSTU, TRTR

## 16.6 Migration considerations

### Hardware

When the ARCH compiler option is used, the resulting program can only be run on the architecture level specified by the option, or higher. The LP64 option requires ARCH(5) or higher. The default is ARCH(5).

When the DFP option is used, the resulting program can only be run on the hardware that has the hardware Decimal Floating Point instructions.

### Software

The SQL option requires DB2 to be installed on the system. This is needed both during compilation and execution of the program. The compiler supports interaction with the DB2 V7 (and up) SQL statement coprocessor.

## 16.7 Prelnit tracing

Prelnit Tracing offers new support to help diagnose problems with a commonly-used interface to Language Environment.

To help Language Environment Prelnit exploiters understand what sequence of events occurred, which services were used (storage management and so on), and what error indications have resulted when the interface is used, z/OS LE V1R9 adds the following:

- ▶ Trace support to Prelnit
- ▶ IPCS support for Prelnit trace and control blocks

This support is provided to improve diagnostics for Prelnit Applications and allow programmers to quickly debug problems, thus avoiding ambiguities in determining the flow of control within the code.

The support provides users with additional diagnostic information that will assist them with debugging Prelnit problems. It is now possible to use:

- ▶ The LEDATA IPCS **verbexit** command to display information about Prelnit environments and their trace tables
- ▶ The Prelnit Trace to determine the sequence of Prelnit calls that occurred prior to failure

## 16.7.1 Migration considerations

A new keyword has been added to the LEDATA IPCS **verbexit** command:

- ▶ PTBL (value) - requests that PreInit information be formatted according to the following values:
  - CURRENT - If current is specified, the PreInit table associated with the current or specified TCB is displayed.
  - address - if an address is specified, the PreInit table located at that address is displayed.)
  - \* - all active and dormant PreInit tables within the current address space are displayed; note that this option is resource-consuming.
  - ACTIVE - all TCBs in the address space have their PreInit tables displayed.
  - PTBL LEDATA output is comprised of:
    - Formatted PreInit control block
    - Formatted PreInit trace table

The LEDATA ALL parameter also generates PreInit information; when used, ALL defaults to PTBL(CURRENT).

## 16.7.2 PreInit tracing characteristics

Tracing is always active; it begins when the PreInit environment is initialized and ends when the environment is terminated. Trace is kept in an in-storage trace table; it is of a fixed size and wraps around when the end is reached.

Formatted trace entries are displayed from oldest to newest. In addition, the active PreInit routine is now identified in the traceback (previously, CEEPIPI/CELQPIPI was always shown) in Figure 16-9.

```
=== > VERBEXIT LEDATA 'PTBL(CURRENT)'  
  
PreInitialization Programming Interface Trace Data  
CEEPIPI Environment Table Entry and Trace Entry :  
  Active CEEPIPI Environment ( Address 25805CB0 )  
  Eyecatcher : CEEIPTB  
  TCB address : 008D1B08  
  
CEEPIPI Environment :  
  Non-XPLINK Environment  
  Environment Type : MAIN  
  Sequence of Calls not active  
  Exits not established  
  Signal Interrupt Routines not registered  
  Service Routines are not active  
  
CEEPIPI Environment Enclave Initialized  
Number of CEEPIPI Table Entries = 2
```

Figure 16-9 LEDATA example

The output in Figure 16-10 on page 270 is from a dump of a 31-bit Prelnit environment. Note that the internal Prelnit table name CEEXPTBL has changed to CEEXIPTB (CELQIPTB for 64-bit), and the customer Prelnit table name remains as CEEXPTBL.

If there are service routines specified in the Prelnit driver, then their names and routine addresses will be displayed.

```

CEEIPI Table Entry Information :
  CEEIPI Table Index 0 ( Entry 1 )
  Routine Name = HLLCRTN
  Routine Type = C/C++
  Routine Entry Point = A5810B38
  Routine Function Pointer = A5810CC0
  Routine Entry is Non-XPLINK
  Routine was loaded by Language Environment
  Routine Address was resolved
  Routine Function Descriptor was valid
  Routine Return Code = 0
  Routine Reason Code = 0
  
```

Figure 16-10 Example of a table

As depicted in Figure 16-11, information about each routine in the Prelnit table is displayed in two ways, as follows:

- ▶ Each field formatted, flag meanings interpreted,
- ▶ Raw storage dump

Empty rows in the Prelnit Table are not displayed.

```

Entry of routine in CEEIPI Table for Index 0 ( 25805DB8 )

+000000 25805DB8 A5810CC0 25811B30 80000000 00000000
00000000 00000000 00000000 00000000
|va...a.....|
+000020 25805DD8 00000000 00000000 00000000 A5810B38
00000003 258117C8 00000003 25810B38
|.....va.....a.H....a..|
+000040 25805DF8 A5810B38 000014C8 C8D3D3C3 D9E3D540
00000000 00000000 00000000 00000000
|va....HHLLCRTN .....|

CEEIPI Table Index 1 ( Entry 2 ) not in use.
  
```

Figure 16-11 Raw storage dump of a table

As depicted in Figure 16-12 on page 271, trace entries contain some input/output parameters for each service call, and each service return code value is displayed along with a meaningful explanation.

```

Entry of routine in CEEPIPI Table for Index 0 ( 25805DB8 )

+000000 25805DB8 A5810CC0 25811B30 80000000 00000000
                00000000 00000000 00000000 00000000
                |va...a.....|
+000020 25805DD8 00000000 00000000 00000000 A5810B38
                00000003 258117C8 00000003 25810B38
                |.....va.....a.H.....a..|
+000040 25805DF8 A5810B38 000014C8 C8D3D3C3 D9E3D540
                00000000 00000000 00000000 00000000
                |va.....HLLCRTN .....|

CEEPIPI Table Index 1 ( Entry 2 ) not in use.

```

Figure 16-12 A trace table entry

As can be noticed in Figure 16-13 and Figure 16-14, the sample application issued an ABEND during the last CALL\_MAIN, so the return, reason and feedback codes as well as the service return code were not set in the trace entry.

```

CEEPIPI Trace Table Entries :
Call Type = INIT_MAIN
PIPI Driver Address = A5800A82
Load Service Return Code = 0
Load Service Reason Code = 0
Most Recent Return Code = 0
Most Recent Reason Code = 0
An ABEND will be issued if storage can not be obtained
PreInit Environment will not allow EXEC CICS commands
Service RC = 0 :A new environment was initialized

```

Figure 16-13 CEEPIPI trace table entry

Figure 16-14 shows the CEEPIPI trace table entry follow-on.

```

Call Type = DELETE_ENTRY
Routine Table Index = 1
Routine Name = HLLCOBOL
Routine Address = A5812E20
Service RC = 0 :The routine was deleted from the
PreInit table.

Call Type = CALL_MAIN
Routine Table Index = 0
Enclave Return Code = 0
Enclave Reason Code = 0
Routine Feedback Code = 0000000000000000
Service RC = 0 :The environment was activated and
the routine called.

```

Figure 16-14 CEEPIPI trace table entry follow-on

## 16.8 DLL diagnostics

z/OS V1R9 Language Environment DLL diagnostics support is being provided to help reduce the amount of time required for debugging DLL failures, which can be difficult to diagnose because of error messages that are too general, and a lack of DLL failure diagnostics.

This support provides the following:

- ▶ A new C/C++ environment variable that allows users to tell the z/OS XL C/C++ Run-Time Library how to handle DLL errors.
- ▶ A new Language Environment control block chain that will contain DLL failure diagnostics.

The new environment variable is `_EDC_DLL_DIAG`. It indicates whether additional DLL diagnostic information should be generated upon failure for the following DLL functions (it has no effect on implicit DLLs):

- ▶ `dllload()`
- ▶ `dllqueryfn()`
- ▶ `dllqueryvar()`
- ▶ `dllfree()`
- ▶ `dlopen()`
- ▶ `dlsym()`
- ▶ `dlclose()`

`_EDC_DLL_DIAG` can take the different possible values:

<b>MSG</b>	Issues DLL error messages to the Language Environment message file.
<b>TRACE</b>	Same as MSG, but also calls the <code>ctrace()</code> function to produce a traceback for each error.
<b>SIGNAL</b>	Same as TRACE, but also signals a condition for each error's feedback code.
<b>QUIET</b>	Turns off all <code>_EDC_DLL_DIAG</code> error diagnostics. This is the default setting.

**Note:** `_EDC_DLL_DIAG` values must be specified in upper case letters in order to be recognized.

`_EDC_DLL_DIAG` also provides improved error messages:

Several improved error messages can be issued by `_EDC_DLL_DIAG`. The error messages that were available in previous releases via `perror()` and `strerror()` will continue to be available with `perror()` and `strerror()`.

For example, `_EDC_DLL_DIAG=MSG` and when `dllload()` is attempted, but the module is not found, the following message is issued by `_EDC_DLL_DIAG`:

```
CEE3501S The module module-name was not found.
```

And the following message is issued by `perror()`:

```
EDC5205S DLL module not found.
```



Figure 16-15 shows a few of the possible ways to set the `_EDC_DLL_DIAG` environment variable.

```
Using the setenv() C/C++ API:
    setenv("_EDC_DLL_DIAG","MSG",1)
From the Unix File System:
    export _EDC_DLL_DIAG=TRACE
From the console using RTO Parmlib:
    SETCEE ceedopt,ENVAR("_EDC_DLL_DIAG=SIGNAL")
```

Figure 16-15 `_EDC_DLL_DIAG` Examples

### CEEDLLF – DLL Failure Control Block

DLL Failure Control Block is a new Language Environment control block containing failure diagnostics for an implicit or explicit DLL failure:

- ▶ It contains diagnostics for up to 10 of the most recent DLL failures are available in circular chain of CEEDLLF control blocks.
- ▶ It is language-neutral, and available to all Language Environment languages that support DLLs.

Because the CEEDLLF chain is anchored off the Common Anchor Area (CAA), which is thread-based, each thread can have its own CEEDLLF chain.

### CEECAA\_CCEEDLLF

This is the anchor for the CEEDLLF chain. It contains the address of the most recent DLL failure's CEEDLLF and is located at offset 996 (x3E4) in the 31-bit CAA, and 648 (x288) for the 64-bit CAA.

The CEEDLLF chain is not allocated until the first DLL error. Therefore, CEECAA\_CCEEDLLF will be zero until the first DLL error.

Figure 16-16 shows some of the CEEDLLF Fields.

CEEDLLF_SERVICE	DLL service that failed (load, query function, free, and so on)
CEEDLLF_REFERENCE_TYPE	DLL reference type (explicit or implicit)
CEEDLLF_LOAD_TYPE	Load type attempted (MVS, UNIX File System, or both)
CEEDLLF_PREV	Pointer to previous CEEDLLF in chain
CEEDLLF_NEXT	Pointer to next CEEDLLF in chain
CEEDLLF_FBTKOK	Feedback token for this failure
CEEDLLF_DLL_NAME	Pointer to DLL name
CEEDLLF_SYMBOL_NAME	Pointer to DLL function/variable name
CEEDLLF_DLL_NAME_LEN	Length of DLL name
CEEDLLF_SYMBOL_NAME_LEN	Length of DLL function/variable name
CEEDLLF_RETCODE1	Unix File System load return code
CEEDLLF_RSNCODE1	Unix File System load reason code
CEEDLLF_RETCODE2	MVS load return code
CEEDLLF_RSNCODE2	MVS load reason code

Figure 16-16 Some of the CEEDLLF fields

## 16.8.1 Language Environment IPCS support

The thread-specific control blocks section of VERBX LEDATA contains CEEDLLF diagnostic information (formatted after the CAA).

Figure 16-17 is an example of a formatted CEEDLLF control block.

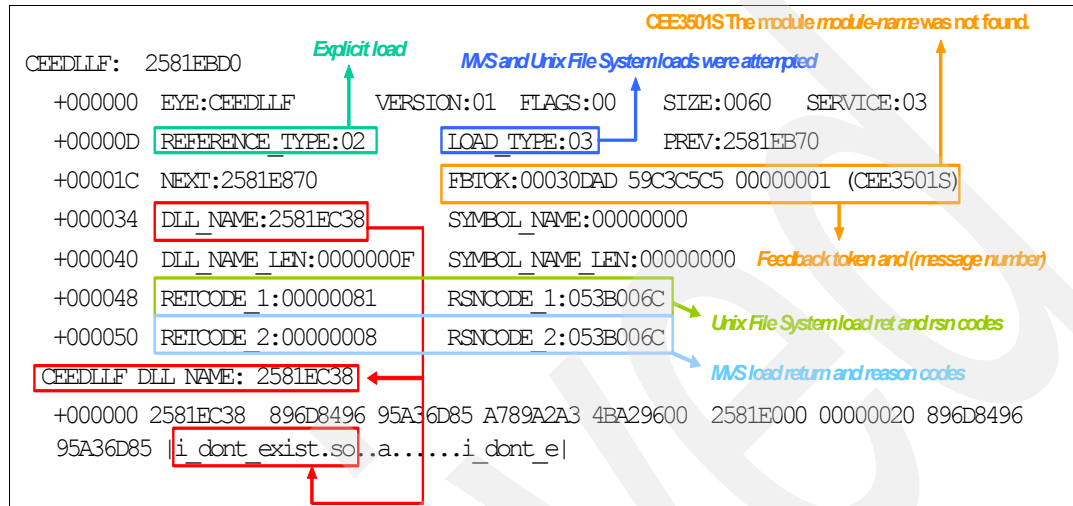


Figure 16-17 Language Environment IPCS support

Only in-use CEEDLLF control blocks are formatted via VERBX LEDATA. In other words, the only CEEDLLF control blocks that are formatted are those which have been populated with DLL failure diagnostics. If no DLL failures occurred before the dump was taken, then no CEEDLLF control blocks are formatted.

If a thread has encountered any DLL failures, the thread-specific control blocks section of the Language Environment dump contains diagnostic information in one or more CEEDLLF control blocks. The format is similar to that displayed in IPCS.

## z/OS UNIX System Services

The z/OS Distributed File Service zSeries File System (zFS) is a z/OS UNIX System Services (z/OS UNIX) file system that can be used in addition to the hierarchical file system (HFS). zFS file systems contain files and directories that can be accessed with z/OS UNIX application programming interfaces (APIs). These file systems can support access control lists (ACLs). zFS file systems can be mounted into the z/OS UNIX hierarchy along with other local (or remote) file system types (for example, HFS, TFS, AUTOMNT and NFS).

In this chapter the new functions for z/OS UNIX System Services and small enhancements to zFS are described, as follows:

- ▶ Automove consistency
- ▶ IOEAGFMT batch utility authorization enhancement
- ▶ Concurrent log recovery
- ▶ Improved dynamic grow
- ▶ Improved hang detection

## 17.1 Automove consistency

Shared file system availability is improved through more consistent and predictable shutdown and recovery actions based on the file system AUTOMOVE attribute to provide AUTOMOVE consistency.

The AUTOMOVE attribute, specified on a MOUNT statement in the BPXPRMxx parmlib member or on the **MOUNT** command, controls the shared file system processing for various shutdown and recovery scenarios. The following values can be specified:

<b>AUTOMOVE</b>	The file system is eligible to move to any system.
<b>NOAUTOMOVE</b>	The file system is not eligible to move. This is typically for system-specific file systems.
<b>UNMOUNT</b>	The file system is unmounted during shutdown or recovery processing. Again, this is typically for system-specific file systems.
<b>System list</b>	A list of eligible or ineligible systems to move the file system to is provided.

You can distinguish between the two different types of PFS capabilities for a specified mount mode, either read-write (RDWR) or read-only (RO), and these specifications can now be either sysplex-unaware and sysplex-aware with z/OS V1R9.

- ▶ A file system that is mounted in a mode that a PFS is sysplex-unaware is referred to as a “sysplex-unaware file system”.
- ▶ A file system that is mounted in a mode that a PFS is sysplex-aware is referred to as a “sysplex-aware file system”.

Sysplex-unaware file systems can only be locally accessed by one owner system in the specified mount mode (RDWR or RO). All other systems access the file system through function shipping to the owner.

**Note:** HFS is sysplex-unaware for RDWR mounts.

Figure 17-1 shows a display of a sysplex-unaware file system from a client system named SY1. Note that the file system owner is system SY2, and also note that CLIENT=Y. This indicates a function-shipping client, which is presumably for a sysplex-unaware file system. This is an HFS file system mounted RDWR.

```
D OMVS,F,N=FS1*
BPX0045I 16.50.49 DISPLAY OMVS 796
OMVS      000E ACTIVE          OMVS=(BE)
TYPENAME  DEVICE  -----STATUS-----  MODE  MOUNTED  LATCHES
HFS       26 ACTIVE              RDWR  02/27/2007  L=35
NAME=FS1.HFS                      16.50.23  Q=35
PATH=/fs1
OWNER=SY2      AUTOMOVE=Y  CLIENT=Y
```

Figure 17-1 Display of a sysplex-unaware file system from a client system

Sysplex-aware file systems can be locally accessed by all systems in the specified mount mode.

**Note:** HFS and zFS are sysplex-aware for RO mounts. In z/OS V1R9, zFS is sysplex-aware for RDWR mounts. NFS is sysplex-aware for RDWR and RO mounts.

Figure 17-2 shows a display of a sysplex-aware file system from a client system. It is captured from system SY1. Note that the file system owner is SY2 and that CLIENT=N. This indicates a sysplex-aware file system mounted locally on the client system.

```
D OMVS,F,N=FS1*
BPX0045I 16.53.03 DISPLAY OMVS 800
OMVS      000E ACTIVE          OMVS=(BE)
TYPENAME  DEVICE  -----STATUS-----  MODE  MOUNTED  LATCHES
HFS              26 ACTIVE              READ  02/27/2007  L=35
NAME=FS1.HFS              16.50.23  Q=35
PATH=/fs1
OWNER=SY2      AUTOMOVE=Y  CLIENT=N
```

Figure 17-2 Display of a sysplex-aware file system from a client system

### 17.1.1 Problems with sysplex-aware file systems without the new support

Inconsistencies in the processing of the various shutdown and recovery scenarios based on the AUTOMOVE setting and PFS capabilities resulted in confusing results, impacting desired availability.

The inconsistencies primarily resulted in the various recovery flows, where the automove setting was ignored and treated as AUTOMOVE=YES for sysplex-aware file systems.

**Note:** A documentation APAR OA12251 and a detailed description named USS APAR OA12251 were created in 2005 to document the recovery and shutdown scenarios for USS file system sharing.

#### Shutdown or crash scenarios for read-only file systems

We list several problems with the old behavior mainly based on the inconsistent behavior in different situations.

##### File system mounted read-only with automove set to UNMOUNT

File system mounted on system SC65 with Automove=Unmount, SC65 crashes and the file system is moved to SC70. The automove setting is kept.

System SC65 gets re-IPLed, finds it the file system mounted already, and can work with it with no problems.

A soft shutdown on SC70 unmounts the file system and applications on SC65 using the file system get into trouble and may stop working.

##### File system mounted read-only with automove set to NOAUTOMOVE

File system mounted on SC65 with Automove=No, SC65 crashes and the file system is moved to SC70 with automove set to YES.

Taking down SC70 will move the file system again.

## File system mounted read-only and having an automove include list set

File system mounted on SC65 with Automove=i,SC65.

If SC65 crashes the file system is moved to another system, for example SC70, and the automove setting is changed to YES.

Using **F OMVS,SHUTDOWN** as part of a clean shutdown processing ends up with the problem situation shown in Figure 17-3 and Figure 17-4.

```
BPXM054I FILE SYSTEM OMVS.TEST.LOCAL FAILED TO MOVE.  RET CODE = 00000079,  
RSN CODE = 119E04B6
```

Figure 17-3 Message BPXM054I on trying to move file system with an automove include list

Figure 17-4 shows the even worse follow-on message.

```
BPXI066E OMVS SHUTDOWN COULD NOT MOVE OR UNMOUNT ALL FILE SYSTEMS  
*82 BPXI070E USE SETOMVS ON ANOTHER SYSTEM TO MOVE NEEDED FILE SYSTEMS, THEN  
REPLY WITH ANY KEY TO CONTINUE SHUTDOWN.
```

Figure 17-4 Message BPXI066E on F OMVS,SHUTDOWN if no new file system owner can be found

**Note:** OMVS shutdown is halted due to a read-only file system with a SYSLIST failing to move the file system. Since the owning system, the only system listed in the INCLUDE SYSLIST, is being taken down, there is no way left to move the file system. Hence, they should have just been unmounted, rather than halting the OMVS shutdown processing.

### 17.1.2 New automove enhancements

The solution is to honor the specified AUTOMOVE setting. This has the benefit that recovery and shutdown processing are more predictable and process as desired. Furthermore, the file system availability is increased and administration is easier.

In z/OS V1R6, the support was changed such that a file system mounted with NOAUTOMOVE or a system list were converted to AUTOMOVE if mounted in a sysplex-aware mount mode. This support is now removed and NOAUTOMOVE and System List are honored as specified.

No sysplex-aware and sysplex-unaware characteristics need to be considered any longer.

**Note:** The new support especially implements the requests addressed with marketing request MR0606057538 (“Shared HFS behavior with various flavors of AUTOMOVE and sysplex-awareness should be consistent.”) in 2005.

The following shutdown and recovery scenarios are discussed in detail:

- ▶ “Soft shutdown processing” (OMVS Shutdown and F BPXOINIT,SHUTDOWN=FILESYS or FILEOWNER)
- ▶ Member gone recovery on hard failures
- ▶ PFS termination
- ▶ Multi-file system moves using SETOMVS, FROMSYS=

## Soft shutdown processing

Soft shutdowns are commands to clean up USS file system processing which are normally run as part of system shutdown processing:

- ▶ F BPX0INIT,SHUTDOWN=FILESYS
- ▶ F BPX0INIT,SHUTDOWN=FILEOWNER
- ▶ F OMVS,SHUTDOWN

Table 17-1 Soft shutdown processing

Automove setting	Behavior
<b>NOAUTOMOVE</b> or UNMOUNT	An attempt to unmount the file system occurs. The unmount will fail if there are other file systems mounted on it.
AUTOMOVE	Move the file system to any system.
<b>System List</b>	Move the file system to any specified eligible system in the system list. If no new owner can be found, then the file system is unmounted as long as no other file systems are mounted on it.

**Note:** Automount-managed file systems are unmounted by a soft shutdown operation if this system is the file system owner and the file system is not being locally used. If the file system is being accessed by another system, then it is moved to that system. This behavior has not been changed.

Figure 17-5 shows an example for a file system mounted read-only on a z/OS V1R9 system using an automove system list setting.

```
#> sysvar SYSNAME
SC70
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.09.00
#> pwd
/u/hering
#> /usr/sbin/mount -t zfs -ra include,SC70 -f OMVS.HERING.TEST.ZFS test
#> df -v test | grep Aggregate
Aggregate Name : OMVS.HERING.TEST.ZFS
#> df -v test | grep Owner
File System Owner : SC70          Automove=I      Client=N
#> df -v test | grep Include
System List (Include) : SC70
```

Figure 17-5 Mounting a file system read-only using a system list on z/OS V1R9

Just for information, this USS file system sharing environment is a mixture of z/OS V1R8 and z/OS V1R9 systems. However, this fact does not influence the following processing.

Figure 17-6 on page 280 demonstrates that this file system is unmounted now if a soft shutdown is performed on system SC70.

**Note:** Using **F BPXONIT,SHUTDOWN=FILESYS** in a USS sysplex sharing environment is only useful if you want to continue using USS and mount file systems again.

If you want to perform a system shutdown, then **F BPXOINIT,SHUTDOWN=FILEOWNER** or **F OMVS,SHUTDOWN** are the preferable choices.

```

SC70:  F BPXOINIT,SHUTDOWN=FILESYS
...
SC65:  IOEZ00044I Aggregate OMVS.HERING.TEST.ZFS attached successfully.
SC70:  IOEZ00416I Aggregate OMVS.HERING.TEST.ZFS moved to system SC65 at
        shutdown.
...
SC70:  BPXM044I BPXOINIT FILESYSTEM SHUTDOWN COMPLETE.
SC65:  IOEZ00048I Detaching aggregate OMVS.HERING.TEST.ZFS
SC70:  D OMVS,F,N=OMVS.HERING.TEST.ZFS
SC70:  BPX0042I 23.04.21 DISPLAY OMVS 505
        OMVS      0011 ACTIVE          OMVS=(9A)
        OMVS.HERING.TEST.ZFS NOT FOUND

```

Figure 17-6 Performing soft shutdown on system SC70 unmounts the file system

The sequence of zFS messages shown in Figure 17-6 is a result of zFS internal processing based on the information received from OMVS. The messages are listed simply for reference. The important fact is that the file system is unmounted as the result of a soft shutdown.

**Note:** A direct benefit of this is that you now can have an automove inclusion list containing all systems currently using the same version root mounted as read-only. As soon as the last system using the file system leaves the sysplex and the file system becomes obsolete, it is unmounted automatically.

This only works easily if you avoid mounting any file systems directly on a version root file system. We suggest that you replace mount point directories by symlinks pointing to a central location directly or close under the sysplex root file system.

### Member gone recovery on hard failures

Table 17-2 lists and describes the new hard failure behavior. Hard failure normally means a system failure.

Table 17-2 Member gone recovery and partition recovery on hard failures

Automove setting	Behavior
NOAUTOMOVE	The file system becomes UNOWNED. The file system remains unowned until the prior owner system restarts.
UNMOUNT	File system is unmounted, as well as all file systems mounted within it.
AUTOMOVE	Move the file system to any system. If no new owner, then the file system becomes UNOWNED.
System List	Move the file system to any specified eligible system in the system list. If no new owner, then the file system as well as all file systems mounted within it are unmounted.



**Note:** For automount-managed file systems, which are normally mounted with Automove=YES, no attempt to take over is performed if the file system is not being used locally. If no system is locally using the file system, then the file system is unmounted. If another system is referencing the file system, it is moved to that system. This behavior has not been changed.

## PFS termination

PFS termination means one of the following situations:

- ▶ Stopping a PFS, for example zFS: **F OMVS,STOPPFS=ZFS**
- ▶ Generally taking down a PFS like NFS client: **C NFSC**
- ▶ PFS has hard failure and terminates

Table 17-3 lists and describes PFS termination recovery processing.

Table 17-3 PFS termination recovery processing

Automove setting	Behavior
NOAUTOMOVE	The file system is unmounted, as well as all file systems mounted within it.
UNMOUNT	File system is unmounted, as well as all file systems mounted within it.
AUTOMOVE	Move the file system to any system. If no new owner is found, then the file system as well as all file systems mounted within it are unmounted.
System List	Move the file system to any specified eligible system in the system list. If no new owner is found, then the file system as well as all file systems mounted within it are unmounted.

**Note:** The prior behavior for Automove=UNMOUNT was to move the file system to another owner. In case of NOAUTOMOVE or when using a system list, the automove setting was changed to AUTOMOVE as in the situations discussed.

Assume file system OMVS.HERING.TEST.ZFS is mounted the same way as shown in Figure 17-5 on page 279. Stopping zFS now will force unmounting of the file system because no new owner can be found. This is shown in Figure 17-7.

```

SC70:  F OMVS,STOPPFS=ZFS
SC70:  *054 BPXI078D STOP OF ZFS REQUESTED. REPLY 'Y' TO PROCEED. ANY OTHER
        REPLY WILL CANCEL THIS STOP.
SC70:  R 54,Y
SC70:  IEE600I REPLY TO 054 IS;Y
SC70:  BPXF063I FILE SYSTEM OMVS.HERING.TEST.ZFS WAS SUCCESSFULLY UNMOUNTED.
SC63:  BPXF063I FILE SYSTEM OMVS.HERING.TEST.ZFS WAS SUCCESSFULLY UNMOUNTED.
SC64:  BPXF063I FILE SYSTEM OMVS.HERING.TEST.ZFS WAS SUCCESSFULLY UNMOUNTED.
...
SC65:  BPXF063I FILE SYSTEM OMVS.HERING.TEST.ZFS WAS SUCCESSFULLY UNMOUNTED.

```

Figure 17-7 Stopping zFS on system SC70 unmounts the file system

## Moving multiple file systems to a specific target system

This is done with the MVS system command SETOMVS as shown in Figure 17-8.

```
SETOMVS FILESYS, FROMSYS=SY1, SYSNAME=SY2
```

Figure 17-8 Moving multiple file systems to a specific target system

Table 17-4 shows the new behavior for moving multiple file systems.

Table 17-4 PFS termination recovery processing

Automove setting	Behavior
NOAUTOMOVE	Move is not attempted.
UNMOUNT	Move is not attempted.
AUTOMOVE	Move is attempted to the target system.
System List	Move is attempted to the target system; the system list is ignored.

**Note:** The prior behavior for Automove=UNMOUNT was to move the file system to another owner. In case of NOAUTOMOVE, or when using a system list, the automove setting was changed to AUTOMOVE as in the situations discussed.

### Moving multiple file systems to any target system

This is done with the MVS system command SETOMVS as shown in Figure 17-9.

```
SETOMVS FILESYS, FROMSYS=SY1, SYSNAME=*
```

Figure 17-9 Moving multiple file systems to any target system

Table 17-5 lists and describes the new behavior for moving multiple file systems.

Table 17-5 PFS termination recovery processing

Automove setting	Behavior
NOAUTOMOVE	Move is not attempted.
UNMOUNT	Move is not attempted.
AUTOMOVE	Move the file system to any system.
System List	Move is attempted to an eligible target systems only.

**Note:** The prior behavior for Automove=UNMOUNT was to move the file system to another owner. In case of NOAUTOMOVE, or when using a system list, the automove setting was changed to AUTOMOVE as in the situations discussed.

### 17.1.3 Migration and coexistence considerations

The following facts need to be taken into account or simply accepted.

Review the BPXPRMxx ROOT and MOUNT statements and make sure that the AUTOMOVE value is the desired behavior for sysplex-aware file systems.

The new support only applies to z/OS V1R9 systems. As a result, you will have a mixture of behaviors until z/OS V1R9 is the lowest release level that is used.

- ▶ If there is a file system mounted read-only with the UNMOUNT attribute, prior releases will ignore the setting, but z/OS V1R9 will honor it.
- ▶ In case of a file system mounted read-only with NOAUTOMOVE on a system suffering a hard failure, prior releases will take ownership and convert to the automove setting to YES.

## 17.2 zFS small enhancements

z/OS V1R9 has small enhancements and simplifications that are added to zFS using APARs to rollback to previous releases what was requested. The z/OS V1R9 enhancements are:

- ▶ zFS authorization to create a zFS aggregate is not the same as HFS
  - User is required to be a superuser to format a zFS aggregate
  - With R9, allow ALTER authority to format a zFS aggregate (rollback to R8 and R7)
- ▶ Concurrent log recovery
- ▶ Improved dynamic grow
- ▶ Improved hang detection
- ▶ zFS R/O mount sometimes fails due to need to replay log
  - zFS needs to handle R/O mount log recovery automatically
  - With R9, allow an option to handle R/O mounts

### 17.2.1 IOEAGFMT and IOEAGSLV authorization

Prior to z/OS V1R9, in order to run the IOEAGFMT format utility to format a zFS aggregate, you need to be a superuser with either one of the following:

- ▶ A UID=0
- ▶ READ authority to the SUPERUSER.FILESYS.PFSCCTL profile in the UNIXPRIV class.

With z/OS v1R9, the new check being made does not add any additional protection. The idea is to ensure that users who could format a zFS aggregate would also need some UNIX authority because they could overwrite existing data. But anyone who has UPDATE authority to the data set can do that anyway. So, the UID=0 check during format is was really not needed.

Furthermore, this behavior was unwanted because when creating an HFS data set, you need ALTER authority to the data set profile only.

The zFS behavior is now changed and makes it more like HFS. The enhancement requires ALTER access for the authorization to use the IOEAGFMT and IOEAGSLV utilities. Table 17-6 lists and describes the associated APAR numbers to make this change available on previous releases.

Table 17-6 APAR numbers for IOEAGFMT and IOEAGSLV authorization enhancement

z/OS release	PTF release	APAR number
z/OS V1R7 and z/OS V1R8	370 and 380	OA18981

z/OS release	PTF release	APAR number
z/OS V1R9	390	OA20613

### APAR OA20613 for z/OS V1R9

This APAR removes the authorization check for a UID using the IOEAGFMT and IOEAGSLV utilities, and the user does not have to have a UID=0.

For the IOEAGFMT utility, the user must have ALTER authority to the VSAM LDS, or be UID 0, or have READ authority to the SUPERUSER.FILESYS.PFSCCTL profile in the RACF UNIXPRIV class. In fact, UPDATE authority to the VSAM LDS is sufficient for format, but zFS will not be able to set the zFS bit in the catalog unless the issuer has ALTER authority.

For the IOEAGSLV utility, the user needs UPDATE authority for the specified VSAM LDS, or be UID 0, or have READ authority to the SUPERUSER.FILESYS.PFSCCTL profile in the RACF UNIXPRIV class.

Set the zFS bit in the catalog, which can be displayed with IDCAMS or the TSO **LISTCAT ALL** command as seen in Figure 17-10 (shown before setting the bit) and Figure 17-12 on page 285 (shown after setting the bit).

```
ioezadm define -aggregate omvs.testfmt.zfs -storageclass openmvs -cylinders 1 1
IOEZ00248I VSAM linear dataset omvs.testfmt.zfs successfully created.
listcat entries('omvs.testfmt.zfs') all
CLUSTER ----- OMVS.TESTFMT.ZFS
...
  ATTRIBUTES
  KEYLEN-----0          AVGLRECL-----0          BUFSPACE-----
  RKP-----0            MAXLRECL-----0          EXCPEXIT-----
  SHROPTNS(3,3)  RECOVERY  UNIQUE           NOERASE        LINEAR
  UNORDERED      NOREUSE   NONSPANNED
...
```

Figure 17-10 Defining a new zFS aggregate and listing LDS attributes

After defining the aggregate, you can format it using IOEAGFMT (in JCL) or in non-superuser mode (using an external link from USS). This is shown in Figure 17-11 on page 285.

```

$> id
uid=888(HERING) gid=2(SYS1) groups=1047(USSTEST)
$> swsu ln -e IOEAGFMT bin/zfsformat
$> zfsformat -aggregate omvs.testfmt.zfs -compat
IOEZ00004I Formatting to 8K block number 90 for primary extent of
OMVS.TESTFMT.ZFS.
IOEZ00005I Primary extent loaded successfully for OMVS.TESTFMT.ZFS.
IOEZ00535I *** Using initialempty value of 1.
*** Using 89 (8192-byte) blocks
*** Defaulting to 13 log blocks(maximum of 1 concurrent transactions).
IOEZ00327I Done. OMVS.TESTFMT.ZFS is now a zFS aggregate.
IOEZ00048I Detaching aggregate OMVS.TESTFMT.ZFS
IOEZ00071I Attaching aggregate OMVS.TESTFMT.ZFS to create HFS-compatible file
system
IOEZ00074I Creating file system of size 720K, owner id 888, group id 2,
permissions x1ED
IOEZ00048I Detaching aggregate OMVS.TESTFMT.ZFS
IOEZ00077I HFS-compatibility aggregate OMVS.TESTFMT.ZFS has been successfully
created

```

Figure 17-11 Formatting the zFS aggregate without being authorized

The parameters used in Figure 17-11 for zfsformat are the same that you would use with IOEAGFMT JCL.

#### Notes:

- ▶ Utility swsu is used to run exactly one command in superuser mode if you have read access to FACILITY profile BPX.SUPERUSER. For more information about this topic, refer to *z/OS Distributed File Service zSeries File System Implementation z/OS V1R7*, SG24-6580.
- ▶ The value x"1ED" is the well-known o755.

Figure 17-12 shows that IOEAGFMT also set on the zFS bit in the catalog. zFS would also try to set the bit during mount processing if it is not set already.

```

listcat entries('omvs.testfmt.zfs') all
CLUSTER ----- OMVS.TESTFMT.ZFS
...
ATTRIBUTES
  KEYLEN-----0          AVGLRECL-----0          BUFSPACE-----
  RKP-----0            MAXLRECL-----0          EXCPEXIT-----
  SHROPTNS(3,3)  RECOVERY  UNIQUE              NOERASE          LINEAR
  UNORDERED      NOREUSE   NONSPANNED          ZFS
...

```

Figure 17-12 Listcat output showing the LDS is marked as a zFS aggregate

## zFS utility considerations

The utility IOEAGFMT now allows you to create and format a zFS aggregate in a job if you have ALTER access to the data set profile. With an external link like zfsformat, you can do the same from a USS shell environment.

The other zFS utility, IOEAGSLV, the salvager, is also included in this change. IOEAGSLV can now be used successfully if you have UPDATE authority for the data set profile. There is no need anymore for running in superuser mode or having READ authority to the SUPERUSER.FILESYS.PFSCCTL profile in the UNIXPRIV class.

Nevertheless, note that even when only verifying a zFS aggregate using option `-verifyonly`, IOEAGSLV still accesses the aggregate exclusively—so, the aggregate cannot be mounted.

Figure 17-13 lists a JCL sample.

```
//SALVAGE EXEC PGM=IOEAGSLV,REGION=OM,  
// PARM=(' -aggregate OMVS.TESTFMT.ZFS -verifyonly')  
//SYSPRINT DD SYSOUT=*
```

Figure 17-13 Using IOEAGSLV in a job

Figure 17-14 shows the successful case.

```
Verifying OMVS.TESTFMT.ZFS  
Processed 1 vols 6 anodes 1 dirs 0 files 0 acfs  
Done. OMVS.TESTFMT.ZFS checks out as zFS aggregate.
```

Figure 17-14 Output of a successfully running salvager verification

Finally, Figure 17-15 shows the unsuccessful case because the zFS aggregate OMVS.HERING.TEST.ZFS is mounted.

```
IKJ56225I DATA SET OMVS.HERING.TEST.ZFS ALREADY IN USE, TRY LATER+  
IKJ56225I DATA SET IS ALLOCATED TO ANOTHER JOB OR USER  
IEF237I JES2 ALLOCATED TO SYSOUT  
IOEZ00003E While opening minor device 1, could not open dataset  
OMVS.HERING.TEST.ZFS.
```

Figure 17-15 Output if salvager is unable to access the zFS aggregate

## 17.2.2 Concurrent log recovery

A new IOEFSPRM keyword, `recovery_max_storage=`, is available to indicate the maximum amount of zFS address space storage to use for concurrent log recovery during multiple concurrent aggregate mounts (attaches).

This allows multiple concurrent mounts to occur when sufficient storage is available for multiple concurrent log recovery processing. The specifications are:

Default Value: 256M  
Expected Value: A number in the range of 128M - 512M.  
Example: `recovery_max_storage=128M`

## 17.2.3 Improved dynamic grow

A new specification on the `zfsadm define` command can be used to improve the possibility that a dynamic grow succeeds.

```
zfsadm define -cylinders primary [secondary]
```

This new option specifies the primary and, optionally, the secondary allocation size for the VSAM LDS in cylinders. The VSAM LDS must have a secondary allocation size specified, if you want to use dynamic grow.

The dynamic grow option for an aggregate can be specified using the following an aggrgrow option, as follows:

- ▶ IOEFSPRM member
- ▶ MOUNT commands
- ▶ ISHELL specification on mounts

## 17.2.4 Improved hang detection

The zFS hang detector monitors the current location of the various tasks processing in zFS. At a set interval, the hang detector thread wakes up and scans the current user requests that have been called into zFS. The hang detector processes this list of tasks and notes various pieces of information that allow it to determine the location of the task.

When the hang detector determines that a task has remained in the same location for a predefined period of time, it attempts to determine why it is hung and if so, the hang detector flags the task as a potential hang and either issues message IOEZ00524I and produces a dump, or issues IOEZ00547I to the console.

If, on a subsequent iteration, the hang detector recognizes that this task has finally progressed, it will DOM the message (remove it from the console). If the message is removed, it means that the hang condition cleared.

## 17.2.5 Hang detection messages

Messages IOEZ00524I and IOEZ00547I are also issued and cleared when a slowdown occurs. This is not an indication of a real hang, but instead that things are progressing slowly because of a stressful workload or some other issue. In this case, you can discard the dump.

Continually monitor for the following messages:

IOEZ00524I zFS has a potentially hanging thread caused by: UserList where: UserList is a list of address space IDs and TCB addresses causing the hang.

IOEZ00547I zFS has a potentially hanging XCF request on systems: Systemnames where: Systemnames is the list of system names.

## 17.2.6 Analyzing hang conditions

To start investigating, enter the **D OMVS,W** command to check the state of sysplex messages and waiters. Message IOEZ00547I (hanging XCF request) can indicate an XCF issue. Check any outstanding message that might need a response to determine if a system is leaving the sysplex or not (for example, IXC402D). This might look like a zFS hang until that message gets a response. Then do the following:

- ▶ Enter the **F ZFS,QUERY,THREADS** command to determine if any zFS threads are hanging and why.

**Note:** The type and amount of information displayed as a result of this command is for internal use and can vary between releases or service levels.

- ▶ Enter the **D A,ZFS** command to determine the zFS ASID.
- ▶ Enter the **F ZFS,QUERY,THREADS** command at one- to two-minute intervals for six minutes.
- ▶ Interrogate the output for any user tasks (tasks that do not show the zFS ASID) that are repeatedly in the same state during the time you requested **F ZFS,QUERY,THREADS**. If there is a hang, this user task will persist unchanged over the course of this time span. If the information is different each time, there is no hang.
- ▶ Verify that no zFS aggregates are in the QUIESCED state by checking their status using the **zfsadm lsaggr** or **zfsadm aggrinfo** command. For example, quiesced aggregates display as follows:

```
DCESVPI:/home/susvpi/> zfsadm lsaggr

IOEZ00106I A total of 1 aggregates are attached
SUSVPI.HIGHRISK.TEST                               DCESVPI    R/W QUIESCE
DCESVPI:/home/susvpi/> zfsadm aggrinfo
IOEZ00370I A total of 1 aggregates are attached.
SUSVPI.HIGHRISK.TEST (R/W COMP QUIESCED): 35582 K free out of total 36000
DCESVPI:/home/susvpi/>
```

Resolve the QUIESCED state, continuing to determine if there is a real hang condition. The hang condition message can remain on the console for up to a minute after the aggregate is unquiesced.

**Note:** Message IOEZ00581E appears on the system that contains at least one zFS aggregate that is quiesced. There is a time delay between when the aggregate is quiesced and when the message appears. When there are no quiesced zFS aggregates on the system, this message is DOMed.

There is also a delay between when the last aggregate is unquiesced and when the message is DOMed. This message is handled by a thread that wakes up every 30 seconds and checks for any quiesced aggregates owned by this system.

It is possible for an aggregate to be quiesced and unquiesced in the 30-second sleep window of the thread and no quiesce message to appear. This message remains if one aggregate is unquiesced and another is quiesced within the 30-second sleep window.

## Final considerations

Finally, if the previous steps do not clear the hang, do one of the following:

- ▶ Enter the **F ZFS,HANGBREAK** command to attempt to break the hang condition. The **F ZFS,HANGBREAK** command posts any threads that zFS suspects are in a hang condition with an error and can cause abends and dumps to occur, which you can ignore.

After entering the **F ZFS,HANGBREAK** command, the hang message can remain on the console for up to one minute. When the **F ZFS,HANGBREAK** command completes, it issues message IOEZ00025I.

However, IOEZ00025I does not mean the system cleared the hang. Enter the **F ZFS,QUERY,THREADS** command to check the output for indication the hang is clear. It is possible that the **F ZFS,HANGBREAK** command can clear the current hang condition only to encounter yet another hang. You may have to enter the **F ZFS,HANGBREAK** command several times.

- ▶ If users are hung in the file system, forcefully unmount the file system by entering the **F ZFS,ABORT** command.



## 17.2.7 z/OS V1R9 enhancements

With z/OS V1R9, a new IOEFSPRM specification can be used to improve hang detection.

hang\_detection\_interval=45 - default is 45 seconds

This improves potential hangs to avoid reporting false hangs. The hang detection can be turned on and off with a modify command with an operator command, as follows:

```
F ZFS,HANGDETECT,ON - to turn on
F ZFS,HANGDETECT,OFF - to turn off
hang_detection=on is default
```

Archived

## SDSF enhancements

This chapter describes the SDSF support for the REXX programming language, added in z/OS V1R9. Using REXX with SDSF provides a simpler and more powerful alternative to using SDSF in batch.

Today, prior to z/OS V1R9, SDSF batch supports only simple programs that issue action characters and (with the ISFAFD flavor of batch) modify values. With REXX, you can include logic that does things like examine values and make decisions based on the values. You can access almost all of SDSF's function. Using REXX, it is possible to do things that are impossible to do with SDSF interactively.

The following topics are described, along with examples:

- ▶ Issuing SDSF commands
- ▶ Issuing operator commands
- ▶ Issuing SDSF action characters
- ▶ Browsing job output
- ▶ Printing job output
- ▶ Diagnosing errors with SDSF REXX execs

## 18.1 SDSF and the REXX programming language

REXX is an interpreted programming language. Its free format, built-in functions, debugging capabilities, and extensive parsing capabilities make it easy for both beginners and professionals to write REXX programs.

Starting with z/OS V1R9, SDSF allows access to SDSF data and functions using the REXX programming language. Using SDSF REXX support provides a simpler and more powerful alternative to using SDSF in batch.

REXX support includes all of the SDSF panels that are supported interactively, with the exception of SYSLOG and output descriptors (which is displayed with the Q action character from job and output panels). The SDSF REXX support is provided by means of:

- ▶ A dynamic REXX host command environment
- ▶ SDSF REXX environment commands
- ▶ Special SDSF REXX variables

This chapter describes only a subset of the SDSF REXX support, along with examples. For a more detailed description of SDSF REXX support, refer to the following resources:

- ▶ *SDSF Operation and Customization, SA22-7670*
- ▶ An IBM Redbooks publication, *Implementing REXX support in SDSF, SG24-7419*
- ▶ The REXXHELP command from any SDSF interactive panel

Use the REXXHELP command to access SDSF's online help regarding SDSF REXX support. REXXHELP includes links to descriptions of commands, action characters and overtypable columns, which are not included in any other manuals.

### 18.1.1 SDSF REXX and System REXX

SDSF REXX support is also available under System REXX. If you invoke SDSF REXX using System REXX, you need to be aware of the following:

- ▶ You must set up the ISFJESNAME variable, a new special REXX variable, to identify the JES2 subsystem. A new optional parameter, JESNAME, on the SDSF command specifies the JES2 subsystem that SDSF is to process, as follows:

```
SDSF ISFJESNAME
```

- ▶ You must be authorized to invoke SDSF functions from REXX

JESNAME parameter on SDSF command must be protected using SAF with the following resource name:

- ISFCMD.OPT.JESNAME

### 18.1.2 Authorization for SDSF and REXX

You must be authorized to use SDSF from REXX, and you must be authorized to the SDSF functions that you invoke from REXX. In some cases, invoking an SDSF function from REXX when you are not authorized to the function causes the exec to fail and the SDSF session to end.

Using the SDSF function from a REXX exec is protected just as using SDSF interactively is protected, with the same SAF resources and ISFPARMS parameters. Where special REXX

variables correspond to SDSF commands, the authorization for those special variables is the same as for the associated command.

To control which group in ISFPARMS a user is assigned to, you can use either SAF or ISFPARMS. Using SAF is the recommended approach, as it is more dynamic and allows you to assign users to the same group regardless of the environment from which they invoke SDSF (interactive, batch, or REXX). To determine group membership, SDSF checks the SAF resource, as follows:

```
GROUP.group-name.server-name in the SDSF class

RDEFINE SDSF GROUP.group-name.server-name UACC(NONE)
PERMIT GROUP.group-name.server-name CLASS(SDSF) ID(userid or groupid)
ACCESS(READ)
```

## 18.2 Setting up the SDSF host command environment

You invoke the SDSF function with a new host command environment, SDSF, new ISFEXEC and ISFACT commands, and REXX variables. Data and SDSF messages are returned in REXX variables. Accessing the SDSF function with REXX requires use of the following SDSF functions:

- ▶ The **ISFCALLS** command - to add and delete the SDSF host command environment.
- ▶ The **ISFEXEC** command, for SDSF commands such as the commands that access SDSF panels and to issue / and **WHO** commands.
- ▶ The **ISFACT** command, for action characters and overtyping columns.
- ▶ Special REXX variables, to provide function equivalent to other SDSF commands, and for messages and table data and they contain the values for SDSF panels, and allow you to control results.

Before calling any of the SDSF host commands in a REXX program, you must set up the SDSF host command environment. To set up the SDSF host command environment, use the **isfcalls** command. The following call to **isfcalls** dynamically sets up the SDSF host command environment.

```
rc = isfcalls('ON')
```

Once the SDSF host command environment is set up, use **address SDSF** to issue the SDSF environment commands under REXX.

To delete the SDSF host command environment call **isfcalls** as follows:

```
rc = isfcalls('OFF')
```

### Return codes

A return code of 0 indicates that the **isfcalls** command completed successfully. The return codes are:

- 00** Function completed successfully
- 01** Host command environment query failed, environment not added
- 02** Host command environment add failed
- 03** Host command environment delete failed

## 18.2.1 Issuing SDSF commands in a REXX program

After setting up the SDSF host command environment, you can start issuing SDSF commands in a REXX program. To issue SDSF commands under REXX, use the **ISFEXEC** command. **ISFEXEC** allows you to issue SDSF panel commands (such as **ST** or **DA**) and the slash (/) command. In addition, special REXX variables provide functionality equivalent to many other SDSF commands such as **PREFIX**, **FILTER**, or **SORT**. Figure 18-1 shows the syntax of the **ISFEXEC** command.

```
address SDSF "ISFEXEC sdsf-command (options)"
```

Figure 18-1 Syntax of the ISFEXEC command

Where:

An **sdsf-command** can be any SDSF command that accesses an SDSF panel, the slash (/) command, and the **WHO** and **QUERY** commands. The maximum length of commands entered with **ISFEXEC**, including the parameters, cannot exceed 42 characters. The **LOG** and **ULOG** commands are not supported through **ISFEXEC**. See 18.3.2, "Issuing operator commands" on page 298 for a description of how the access data in the **ULOG** panel.

### Data returned from the ISFEXEC command

SDSF panels have a tabular nature. Therefore, the **ISFEXEC** command returns each column's data in a stem variable. A successful call to **ISFEXEC** creates a stem variable for each returned column. The format of each stem variable is **column-name.row-number**. The column name used to create the stem variable for each column is different than the column title that is displayed in the SDSF interactive panels. **ISFEXEC** uses the same name that is used in the **FLD** statement in **ISFPARMS**. For example, in the **ST** panel, the job name is listed under the **JOBNAME** title. The matching **FLD** name for **JOBNAME** is **JNAME**. Thus the stem variable names for the **JOBNAME** column are **JNAME.0**, **JNAME.1**, **JNAME.2**, and so on.

The value for stem variable 0 contains the number of variables returned. This number should be the same for all stem variables returned by a call to **ISFEXEC**. In addition, a special REXX variable named **isfrows** contains the number of rows returned by the last call to **ISFEXEC**, and a special REXX variable name **isfcols** contains the names of all returned columns.

**ISFEXEC** also returns a stem variable named **TOKEN**. A **TOKEN** variable in the format **TOKEN.row-number** is returned for each row. The row token is used to identify a specific row when issuing an action character against the row. See 18.3.3, "Issuing action characters" on page 300 for more information about issuing action characters.

**Tip:** Use the **COLSHELP** command in SDSF under ISPF to see a list of all columns in all SDSF panels and their respective **FLD** names.

### ISFEXEC options

The **ISFEXEC** options is an optional list of options for the command. The closing parenthesis is optional. The following options can be used to provide enhanced functionality to **ISFEXEC** commands:

- |                  |  |
|------------------|--|
| <b>ALTERNATE</b> | Requests the panel's alternate field list.   |
| <b>DELAYED</b>   | Specifies that delayed-access columns be included in the command's output (Use <b>COLSHELP</b> to check which columns are delayed-access columns). |

<b>NOMODIFY</b>	Specifies that row tokens for use in modifying rows should not be returned. Use this to improve performance if you do not intend to modify any values.
<b>PREFIX</b>	Value specifies a prefix for column name and TOKEN variables that are created; use this to ensure that variable names do not conflict between different <b>ISFEXEC</b> commands. The prefix can be up to 24 characters long, and should <i>not</i> begin with ISF.
<b>VERBOSE</b>	Adds diagnostic messages to the <code>isfmsg2</code> stem variable. The messages describe each row variable created by SDSF.

### Special variables that provide support for SDSF commands

The special REXX variables provide support that is equivalent to SDSF commands. The variables are grouped by command type, as follows:

- ▶ SDSF command - Use the following special variables for function that is equivalent to the parameters on the SDSF command.
  - ISFSERVER names the SDSF server and ISFJESNAME names the JES2 subsystem to process
- ▶ Filter commands - Use the following special variables for function that is equivalent to the filter commands, such as FILTER and PREFIX.
  - ISFDEST, ISFFILTER, ISFINPUT, ISFOWNER, ISFPREFIX, and ISFSYSNAME
- ▶ Options commands - Use the following special variables for function that is equivalent to the options commands, such as the SET commands.
  - ISFACTIONS, ISFCONS, ISFDELAY, ISFDISPLAY, ISFINPUT, ISFSCHARS, and ISFTIMEOUT
- ▶ Trace commands
  - ISFTRACE and ISFTRMASK

For example, where the variable is associated with an SDSF command, the parameters for the variable are the same as for the command, with the exception that the parameter is not supported in REXX. Substitute the variable for the command, as shown in this example:

```
SDSF command: PREFIX RJONES*
Variable: isfprefix="RJONES*"
```

## 18.2.2 Special REXX variables

The SDSF special REXX variables provide a mechanism to issue SDSF commands which are not panel commands, through REXX. With the use of these variables, the following can be useful in writing REXX execs:

- ▶ Testing the value of the SDSF special REXX variables after a call to ISFEXEC to get more information about the returned data.
- ▶ Setting the value of some SDSF special REXX variables prior to calling ISFEXEC to control the returned data.

The examples that follow in this chapter use many of the SDSF special REXX variables and are explained prior to the example. A complete list of the SDSF special REXX variables is found in *z/OS SDSF Operation and Customization, SA22-7670*.

Special variables are variables defined by SDSF and are used as follows:

- ▶ Names start with characters ISF; some examples are:
  - `isfprefix=*` which corresponds to the command **PREFIX \***
  - `isfowner=ken` which corresponds to the command **OWNER KEN**
  - `isffilter=jprio gt 5` which corresponds to the command **FILTER GT 5**
- ▶ Used to specify additional options or limit the response
- ▶ Assign value in exec prior to invoking ISFEXEC or ISFACT

## 18.3 Examples of using ISFEXEC

Figure 18-2 shows an example of using ISFEXEC to access the ST panel. This example is explained as follows:

- ▶ In the first line, ISFCALLS adds the REXX host command environment.
- ▶ Before accessing the panel, the `isfprefix` special REXX variable is used to limit the job names to be displayed in the same manner that the SDSF PREFIX command is used.
- ▶ The `isfsort` special REXX variable is used to indicate that the data returned by ISFEXEC is to be sorted by job ID in descending order.
- ▶ The value of `isfcols` is set to a list of only the specific columns required for the program. This requires less storage and improves performance.
- ▶ The `isfdisplay` special REXX variable is used to print the filtering and sorting criteria used by ISFEXEC.
  - The ALTERNATE and DELAYED options are used with the ISFEXEC call in order for the program to be able to access the DATEE and TIMEE columns.
  - Because only data is displayed in this REXX program, and does not modify any data, just add the NOMODIFY parameter to the ISFEXEC call to improve performance.



```

/* REXX */
x = isfcalls('ON')

isfprefix = 'PELEG*'
isfsort   = 'JOBID D'

isfcols = 'JNAME JOBID OWNERID QUEUE RETCODE DATEE TIMEE'

address SDSF "ISFEXEC ST (ALTERNATE DELAYED NOMODIFY"
say isfdisplay
say ' '

do i = 1 to isfrows
  say left(JNAME.i, 8),
      left(JOBID.i, 8),
      left(OWNERID.i, 8),
      left(QUEUE.i, 9),
      left(RETCODE.i, 10),
      left(DATEE.i, 8),
      left(TIMEE.i, 8)
end
x = isfcalls('OFF')

```

Figure 18-2 Example of using ISFEXEC

The output from the example in Figure 18-2 on page 297 is shown in Figure 18-3.

PREFIX=PELEG*	DEST=(ALL)	OWNER=*	SORT=JOBID/D	SYSNAME=
PELEG	TSU29877	PELEG	EXECUTION	2007.122 09:35:55
PELEG	TSU29751	PELEG	PRINT	CC 0000 2007.121 09:24:36
PELEGIDC	JOB29825	PELEG	PRINT	CC 0000 2007.121 18:01:01
PELEGBR5	JOB29771	PELEG	PRINT	CC 0000 2007.121 11:30:55
PELEGBR4	JOB29770	PELEG	PRINT	CC 0000 2007.121 11:30:55
PELEGBR3	JOB29769	PELEG	PRINT	CC 0000 2007.121 11:30:55
PELEGBR2	JOB29768	PELEG	PRINT	CC 0000 2007.121 11:30:55
PELEGBR1	JOB29767	PELEG	PRINT	CC 0000 2007.121 11:30:55
PELEGBR1	JOB29766	PELEG	PRINT	CC 0000 2007.121 11:30:16

Figure 18-3 ISFEXEC example output

**Note:** Use ISFEXEC to access tabular panels (DA, ST, PS, PR, INIT, and so on). You cannot access the LOG panel. The options on ISFEXEC are new options for ISFEXEC, and should not be confused with SDSF command parameters.

### 18.3.1 The WHO and QUERY commands

The SDSF **WHO** and **QUERY** commands are supported with **ISFEXEC**. These commands are different than other SDSF commands in the sense that they do return data, but not in a panel. Therefore, **ISFEXEC** does not create column stem variables for the data returned from the **WHO** and **QUERY** commands. Instead, an SDSF special REXX variable is used. The variable

is named `isfresp`, and is a stem variable. `isfresp.0` contains the number of lines returned from the command. Iterate `isfresp` to access the command's returned data.

Figure 18-4 shows an example of issuing the `WHO` command using `ISFEXEC`, and printing the output to the use terminal.

```
/* REXX */  
  
x = isfcalls('ON')  
  
address SDSF "ISFEXEC WHO"  
  
do i = 1 to isfresp.0  
    say isfresp.i  
end  
  
x = isfcalls('OFF')
```

Figure 18-4 Example of using `ISFEXEC` to issue the `SDSF WHO` command

The output from the example in Figure 18-4 is displayed in Figure 18-5 on page 298.

```
USERID=PELEG  
PROC=IKJACCNT  
TERMINAL=SC38TCA9  
GRPINDEX=1  
GRPNAME=ISFSPROG  
MVS=z/OS 01.09.00  
JES2=z/OS 1.9  
SDSF=HQX7740  
ISPF=N/A  
RMF/DA=NOTACC  
SERVER=YES  
SERVERNAME=SDSF  
JESNAME=JES2  
MEMBER=SC70  
SYSNAME=SC70  
SYSPLEX=SANDBOX  
COMM=NOTAVAIL
```

Figure 18-5 `SDSF REXX` support `WHO` command output example

### 18.3.2 Issuing operator commands

Another command that can be issued using `ISFEXEC` is the `SDSF slash (/)` command. The `SDSF slash (/)` command allows you to enter operator commands to the system. The `W` and `I` prefix parameters of the `slash (/)` command are not supported through `ISFEXEC`. Instead, use the `WAIT` and `INTERNAL` parameters. The maximum length of `slash (/)` commands entered using `ISFEXEC` must not exceed 126 characters.

In an interactive `SDSF` session, the output of the `slash (/)` command is displayed in the `ULOG` panel. The `ULOG` panel is not accessible using the `ISFEXEC` command. To access the output of a `slash (/)` command under `REXX`, a special `REXX` variable is provided by `SDSF REXX`

support. The name of the special REXX variable is `isfulog`. `isfulog` is a stem variable, with `isfulog.0` holding the number of lines that were returned by the last slash (/) command.

You may want to change the name of the EMCS console that SDSF automatically activates when you issue an operator command using the slash (/) command. SDSF provides the **SET CONSOLE** command when in an interactive session. The SDSF REXX equivalent to the **SET CONSOLE** command is the `isfcons` special REXX variable.

### Example

Figure 18-6 on page 299 shows an example of using ISFEXEC to issue a slash (/) command. The sample REXX program receives an operator command as a parameter and prints its output back to the user terminal. In the example, we use the REXX built-in `random` function to generate a random EMCS console name. The `WAIT` parameter is used with the ISFEXEC command to indicate that SDSF should wait the full delay interval before retrieving the responses to the slash (/) command. The delay interval is set using the `isfdelay` special REXX variable. The operator command output is then displayed by iterating the `isfulog` stem variable.

```

/* REXX */

arg opCmd

if opCmd == '' then do
    say 'Please specify an operator command'
    exit 4
end

isfcons = 'PELEG' || random(999)
isfdelay = 2

x = isfcall('ON')

address SDSF "ISFEXEC '/'opCmd'" (WAIT)

do i = 1 to isfulog.0
    say strip(isfulog.i)
end

x = isfcall('OFF')

```

Figure 18-6 Example of issuing an operator command using the slash (/) command

Figure 18-7 shows a sample output of the example in Figure 18-6, when executed with the parameter `D U,IPLVOL`.

```

SC70      2007122  16:47:59.93      ISF031I CONSOLE PELEG574 ACTIVATED
SC70      2007122  16:47:59.93      -D U,IPLVOL
SC70      2007122  16:47:59.94      IEE457I 16.47.59 UNIT STATUS 516
UNIT TYPE STATUS      VOLSER      VOLSTATE
D14E 3390 S           Z19RA1     PRIV/RSDNT

```

Figure 18-7 Slash (/) command example output

### 18.3.3 Issuing action characters

SDSF REXX support allows you to issue action characters against rows returned by SDSF REXX commands. The support is provided by the ISFACT command under **address SDSF**. Using ISFACT, it is also possible to modify overtypable columns in SDSF panels. Figure 18-8 describes the syntax of the ISFACT command.

```
address SDSF "ISFACT command TOKEN("token") PARM(field-value-pairs)"
```

Figure 18-8 Syntax of the ISFACT command

**command** is the same SDSF command, including the same parameters, that was previously issued using ISFEXEC to retrieve the panel's columns.

**token** is the value of the TOKEN stem variable returned by ISFEXEC and matching the row you wish to issue an action character against, or change an overtypable column value in.

**Note:** Tokens returned by ISFEXEC in the TOKEN stem variable are intended to be used shortly after the call to ISFEXEC and by the same caller. Tokens represent jobs at the time they are generated. They should not be saved for later use by a different caller or REXX program. Furthermore, the format of tokens may change incompatibly with service or new releases of SDSF.

The PARM parameter is used to identify the columns you wish to change, and value you wish to update them with. A single call to ISFACT can update multiple columns. The format of the PARM parameter is a **field-name new-value** pair. Where **field-name** is the FLD name, as defined in ISFPARMS, of the field to be updated, and **new-value** is the new value to update the field with. You may specify more than one pair in the PARM parameter.

#### Examples

The first REXX program, shown in Figure 18-9, is an example of how to use ISFACT to issue an action character. In the example, we list all jobs with prefix PELEGBR\* and then issue the P (purge job) action character against each one of them.

```
/* REXX */  
  
x = isfcalls('ON')  
  
isfprefix = 'PELEGBR*'  
address SDSF "ISFEXEC ST"  
  
do i = 1 to isfrows  
    address SDSF "ISFACT ST TOKEN('"TOKEN.i"') PARM(NP P)"  
end  
  
x = isfcalls('OFF')
```

Figure 18-9 Example of issuing an action character using ISFACT

The second REXX program, shown in Appendix 18-10, "Example of modifying an overtypable field using ISFACT" on page 301, is an example of how to use ISFACT to modify an overtypable field. In the example, we list all jobs with prefix PELEG\* in the DA panel and then change their service class to the one specified as a parameter to the REXX program. Note

that SDSF REXX will activate an EMCS console to issue the RESET operator command used to change a job's service class. Therefore, we use the `isfcons` special REXX variable to generate a random console name.

```
/* REXX */  
  
arg srvCls  
  
x = isfcalls('ON')  
  
isfcons = 'PELEG' || random(999)  
isfprefix = 'PELEG*'  
address SDSF "ISFEXEC DA"  
  
do i = 1 to isfrows  
    address SDSF "ISFACT DA TOKEN('TOKEN.i') PARM(SRVCLASS "srvCls")"  
end  
  
x = isfcalls('OFF')
```

Figure 18-10 Example of modifying an overtypable field using ISFACT

### 18.3.4 Browsing job output

SDSF REXX support provides two new action characters, SA and SJA, which are not supported by the interactive SDSF panels. SA is used to allocate all spool data sets associated with the row it was issued against. The SA and SJA action characters can be issued in the DA, H, I, JDS, O and ST panels. For example, when SA is issued next to a job name in the ST panel, all the job's spool data sets are allocated for the REXX program. But if SA is next to a DD name in the job data sets (JDS) panel, only that spool data set is allocated to the REXX program. SJA is used in the same way as SA to allocate only the job's JCL data set. After a spool data set is allocated by SDSF REXX, it can be read using the EXECIO REXX command.

Two special REXX variables are used in support of the SA and SJA action characters:

- isfddname** A stem variable that contains the system-generated DD names returned by the allocation. It is not the same as the application specified DD name (that is contained in the DDNAME stem variable returned by ISFACT). ISFDDNAME.0 contains a count of the number of variables that follow.
- isfddname** A stem variable that contains the application-specified data set name that has been allocated by SDSF. The variables have a one-to-one correspondence with the variables in ISFDDNAME. ISFDDNAME.0 contains a count of the number of variables that follow.

#### Example

The REXX program in Figure 18-11 on page 302 is an example of browsing a job's spool data set. The example program implements a function similar to the `tail` UNIX shell command. The program receives as parameters a job ID and a DD name to print. The program prints only the last lines of the specified DD. The amount of last lines printed can be set by the third parameter of the REXX program. The default is to print the last 20 lines.

In the example, the program first lists the job on the ST panel, by using ISFEXEC. Then, the ? action character is issued next to the job by using ISFACT. Later, the SA action characters are

issued next to the specified DD name. The stem variable **ddname** is created by the first call to ISFACT.

```
/* REXX */

arg jobid taildd lines#

if jobid == '' then do
  say 'Please specify a job id'
  exit 4
end
if taildd == '' then do
  say 'Please specify a DD name'
  exit 4
end
if lines# == '' then
  lines# = 20

x = isfcalls('ON')

isffilter = 'JOBID EQ 'jobid
address SDSF "ISFEXEC ST"

/* get a list of the job's DD names */
isffilter = ''
address SDSF "ISFACT ST TOKEN('TOKEN.1') PARM(NP ?)"

/* remember what row the DD is on */
do i = 1 to isfrows
  if ddname.i == taildd then do
    row# = i
    i = isfrows
  end
end

/* allocate spool data sets */
address SDSF "ISFACT ST TOKEN('TOKEN.row#') PARM(NP SA)"

address TSO "EXECIO * DISKR" isfddname.1 "(STEM sysout. FINIS"

say taildd 'contains a total of' sysout.0 'lines.'
say 'The last' lines# 'lines are:'

/* print the last ## lines of the sysout */
startix = sysout.0-lines#+1
if startix < 1 then
  startix = 1
do i = startix to sysout.0
  say sysout.i
end

x = isfcalls('OFF')
```

Figure 18-11 Example of browsing a job spool data set

### 18.3.5 Printing job output

SDSF provides four action characters for printing a job's output, as explained here.

- ▶ The X action character is used to print all the job's data sets using default settings.
- ▶ The XD action character is used to print all the job's data sets to a specified data set.
- ▶ The XF action character is used to print all the job's data sets to a file already allocated to a specified DD name.
- ▶ The XS action characters is used to print all the job's data sets to a specified SYSOUT file.

All four action characters are also available under SDSF REXX support using the ISFACT command. Under an interactive SDSF session, these action characters open the print panel. Under SDSF REXX, you use special REXX variables as the equivalent to filling in values in the print panel fields.

#### Example

The example in Figure 18-12 on page 304 receives a job ID and a data set name as parameters and prints the job's spool data sets to the specified data set. First, we list the job in the ST panel. Then we use ISFACT to issue the XD command next to the job, on the NP field. At the end of the REXX program, we issue the XDC action characters against the job to close the print data set. Several SDSF special REXX variables are used in the example to control the attributes of the data set the program prints to:

<b>isfprtdisp</b>	Disposition
<b>isfprtrecfm</b>	Record format
<b>isfprtirecl</b>	Logical record length
<b>isfprtblksize</b>	Block size
<b>isfprtspacetype</b>	Space allocation units
<b>isfprtprimary</b>	Primary space allocation
<b>isfprtsecondary</b>	Secondary space allocation
<b>isfprtdsname</b>	Data set name

There are many more special REXX variables that can be used to control the print action characters. For a full list, refer to *SDSF Operation and Customization*, SA22-7670.

```

/* REXX */

arg jobid dsn

if jobid == '' then do
    say 'Please specify a job id'
    exit 4
end
if dsn == '' then
    dsn = userid() || '.SDSFRT.' || jobid || '.LIST'

x = isfcalls('ON')

isffilter = 'JOBID EQ 'jobid
address SDSF "ISFEXEC ST"

isffilter = ''

/* define the output data set attributes */
isfprtdisp      = 'NEW'
isfprtrecfm     = 'FB'
isfprt1recl     = '240'
isfprtblksize  = '3120'
isfprtspacetype = 'CYLS'
isfprtprimary   = '5'
isfprtsecondary = '5'
isfprtdsname   = dsn

/* print the job's output */
address SDSF "ISFACT ST TOKEN('"TOKEN.1"') PARM(NP XD)"
address SDSF "ISFACT ST TOKEN('"TOKEN.1"') PARM(NP XDC)"

x = isfcalls('OFF')

```

Figure 18-12 Example of printing a job's data sets to a data set

## 18.4 Executing REXX execs

You can run an interpreted or compiled REXX program with syscall commands from the following:

TSO/E

MVS batch jobs

z/OS UNIX shells

A program

### 18.4.1 Diagnosing errors in an SDSF REXX exec

SDSF REXX support provides several ways to diagnose errors in SDSF REXX programs.



## Return codes

SDSF REXX commands set a return code in the standard REXX `rc` variable, as do many other REXX commands. It is considered good practice to check the reason code after every call to an SDSF REXX command. A return code higher than zero indicates the command did not complete successfully, and that further diagnosis is required.

## Special REXX variables

When an SDSF REXX command does not complete successfully, it sets two special REXX variables to indicate the problem. The first variable is `isfmsg`, which contains the SDSF short message describing the reason for not completing the request successfully. The second variable is `isfmsg2`, which is a stem variable. `isfmsg2` contains the numbered SDSF messages regarding the error. In case of an error, check these two variables to receive more details about the error.

Another SDSF special REXX variable is `isfdiag`. It contains internal reason codes that are intended for use by IBM service personnel.

## The VERBOSE option

Both ISFEXEC and ISFACT support a VERBOSE optional parameter. When the VERBOSE parameter is specified, SDSF REXX support adds diagnostic messages to the `isfmsg2` stem variable. The messages describe each row that is created by ISFEXEC or ISFACT.

## Trace

It is also possible to activate SDSF's tracing facility under SDSF REXX support. To do so, set the special REXX variable `isftrace` to 'ON'. Use the special REXX variable `isftrmask` to control the mask of traced events.

# 18.5 SDSF migration considerations

There is a new version of SDSF with z/OS V1R9. There are a number of migration issues that need to be considered when using this new version.

## User modifications to SDSF

Although modifying SDSF source modules or macros is not a normally supported technique, some modifications may have been made in this way. The proper way to modify SDSF is to implement a user exit routine.

Beginning with z/OS V1R9, many SDSF source modules and macros that were distributed previously are no longer distributed. You should review any user modifications you have and assess alternatives.

## JES2 and SDSF version levels

Before z/OS V1R9, you had to use the latest level of SDSF with your currently-installed JES2. For example, when migrating to z/OS V1R8, you needed to migrate to the z/OS V1R8 level of SDSF, even if you postponed migrating to the z/OS V1R8 level of JES2.

Starting with z/OS V1R9, SDSF and JES2 are "coupled" in the sense that the release level of your SDSF and JES2 must be the same. This means that:

- ▶ When you migrate to the z/OS V1R9 level of JES2, you must migrate to the z/OS V1R9 level of SDSF.

- ▶ The possible migration scenarios because of SDSF and JES2 being “coupled” are as follows:
  - If you are running the z/OS V1R8 level of JES2 with z/OS V1R9, you must run the z/OS V1R8 level of SDSF.
  - If you are running the z/OS V1R7 level of JES2 with z/OS V1R9, you can run either the z/OS V1R8 or z/OS V1R7 level of SDSF.

## New faces of z/OS

z/OS V1R9 introduces the first steps of a series of ongoing efforts designed to simplify the management, administration and configuration of the system. The goal is to simplify and modernize z/OS management for “zNextGen”, the new generation of IBM System z9 and eServer™ zSeries IT professionals. For example, a new management console powered by IBM Tivoli OMEGAMON® technology is in the works, as well as other enhancements to z/OS to make it fundamentally easier to set up and manage.

These efforts include the creation of the following:

- ▶ New user interfaces (UIs)
- ▶ The enablement of remote access technologies on z/OS
- ▶ Extensions to base operating system components to allow for such management

Marketplace forces, including the “graying” of the z/OS work force, added to the difficulty of attracting and training new skills to this platform due to the perception that the management tools are archaic, and demanded that the platform improve and modernize accessibility in a way that is common to the rest of the IT industry.

This chapter describes the following implementation and changes made with z/OS V1R9:

- ▶ Introduction to the new face of z/OS
- ▶ z/OS V1R9 and the new face of z/OS
- ▶ Common Information Model (CIM)
- ▶ CIM server overview
- ▶ CIM client-to-CIM server access
- ▶ CIM server runtime update and enhancements
- ▶ CIM client API for Java
- ▶ Instrumentation in z/OS
- ▶ Migration and coexistence considerations

## 19.1 Introduction to the new face of z/OS

z/OS users today must master an assortment of user interface (UI) styles: TSO command line, ISPF panels, graphical user interfaces (GUIs), even Web-style UIs. To complete a task such as applying service, they often must interact with different UIs while flipping through a variety of publications. Starting in 2005, a new z/OS management console was introduced to provide a central management for z/OS tasks.

The first release of the console was built on IBM Tivoli OMEGAMON technology and leverages the Tivoli Enterprise Portal console. Health monitoring is the focus of the first release. Using this console, IT staff will be able to monitor the availability of sysplex and system resources and respond to problems. They will also be able to monitor health checks run by the IBM Health Checker for z/OS. Over time, the console will grow to include tasks beyond health monitoring.

### 19.1.1 z/OS ease of use enhancements

To address the skill base for z/OS, a new approach has been developed to increase skills to help less experienced members of the zSeries IT community feel at home in the following areas:

- ▶ z/OS basics

In collaboration with the ITSO, z/OS development is creating an IBM Redbooks publications “basics” library for z/OS. If you are new to z/OS systems programming and have recently assumed the role of system programmer or system analyst after transferring from another area in your organization, this new series of publications, *z/OS Basics*, will help you develop your understanding of the various aspects of the z/OS system.

- ▶ Configuration and operations

The OMEGAMON z/OS Management Console is a no-charge availability monitoring product that includes a GUI for z/OS management and is designed to help the new generation of IT workers. It is designed to help automate, eliminate, and simplify many z/OS management tasks. The OMEGAMON z/OS Management Console helps deliver real-time, health check information provided by the IBM Health Checker for z/OS, and configuration status information for z/OS systems and sysplex resources.

IBM Health Checker for z/OS is a base function for z/OS V1R7. It provides a foundation to help simplify and automate the identification of potential configuration problems before they impact system availability. It compares active values and settings to those suggested by IBM or defined by your installation.

- ▶ Software maintenance

SMP/E V3.4 has been enhanced to provide Internet Service Retrieval. This capability allows you to automate ordering and delivery of PTFs and HOLDDATA. The PTFs and HOLDDATA can be processed in the same job step. This can help eliminate manual tasks currently required for ordering and delivery of IBM PTFs using current methods.

- ▶ Networking

Would you like to dramatically reduce the amount of time that is required to create configuration files? The z/OS Network Security Configuration Assistant is a GUI that you can use to generate the configuration files for both Application Transparent-Transport Layer Security (AT-TLS) and IP Security (IPSec).

The z/OS Network Security Configuration Assistant is a standalone application that runs under the Windows operating system and requires no network connectivity or setup. You can download the GUI from the Communications Server family downloadable tools Web

page. Through a series of wizards and online help panels, you can use the GUI to create both AT-TLS and IPSec configuration files for any number of z/OS images with any number of TCP/IP stacks per image.

► Academic initiative

The IBM Academic Initiative brings together interested colleges and universities, zSeries customer partners, and IBM resources to provide a link between the academic world and the business world and potential future job opportunities. The goal is to increase student awareness of our platform while developing zSeries skills to meet the current and future programmer needs of our zSeries customers.

Membership in the Scholars zSeries Program includes access to a native z/OS system via the Internet for course exercises, system programmer assistance to instructors for setting up and teaching their classes on z/OS, access to zSeries course material, and opportunities for free faculty education. A current focus of the program is working with zSeries customers to help them find schools to partner with to develop zSeries skills.

► A LibraryCenter

The LibraryCenter provides a view of z/OS BookManager® documentation and presents the information in a Windows Explorer format. The LibraryCenter contains documentation for z/OS elements, features, software products and selected z/OS-related IBM Redbooks publications.

There are multiple library centers, one for each release, and also one for z/VM.

► Security

These RACF-based products can help you manage security and monitor compliance:

- Partnership with Vanguard Integrity Professionals, Inc. including the following products: Administrator, Advisor, Analyzer, Enforcer, and SecurityCenter
- IBM Tivoli administration for RACF - lower function alternative

## 19.2 z/OS V1R9 and new faces of z/OS

With z/OS V1R9, various steps have been taken to lay the foundation for simplified access to the z/OS platform.

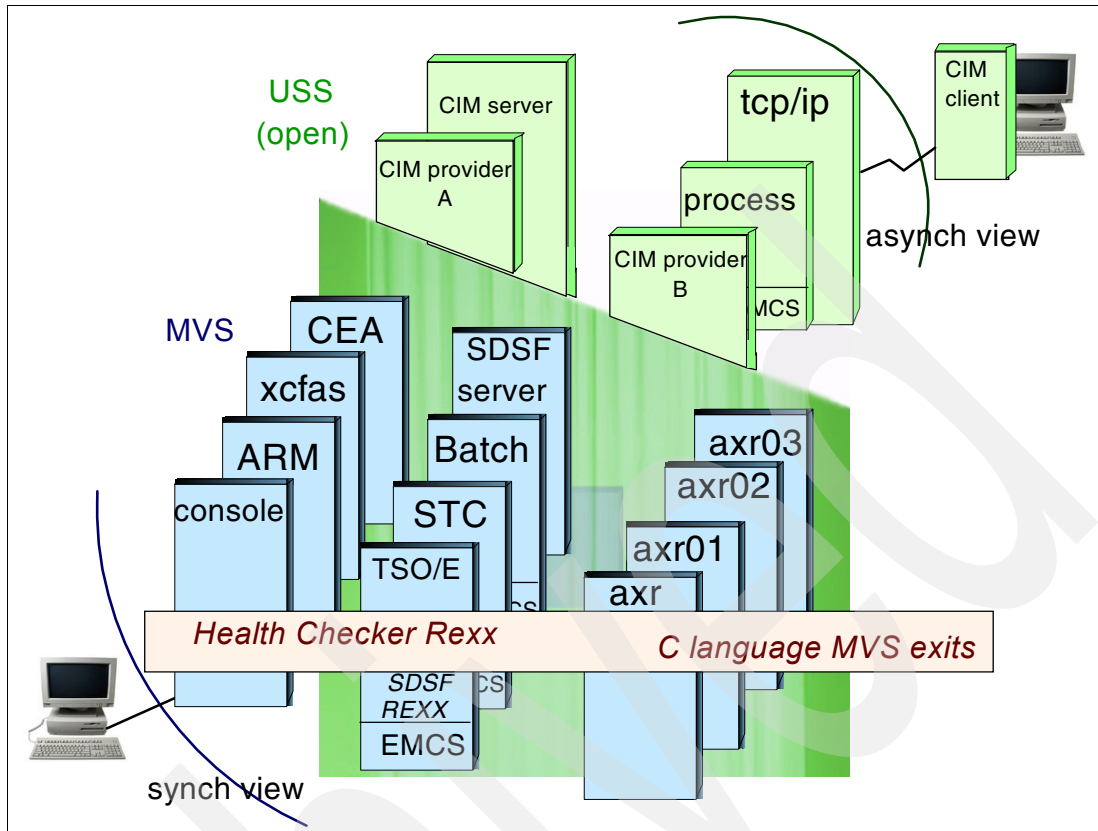


Figure 19-1 New faces of z/OS

These “new faces” are embodied under the cover of different functions in z/OS V1R9—with a set of new tools enhancing the “MVS face” of z/OS, as shown in Figure 19-1, to provide the following new functions.

## 19.2.1 System REXX

System REXX is a z/OS component that allows REXX execs to be executed outside of conventional TSO/E and batch environments. REXX has long been considered one of the fastest development languages for system exit and utilities work on z/OS. The possibilities for exploiting REXX code through the use of System REXX are vast, whether to provide operator assists or to provide an easy way to process files and strings.

The System REXX environment provides a function package that allows a REXX exec to invoke the system commands and to return results back to the invoker in a variety of ways. System REXX execs may be initiated through an assembler macro interface called AXREXX, or through an operator command.

The System REXX address space, AXR, is non-cancelable, but can be terminated by invoking the **FORCE AXR,ARM** command. When the AXR address space terminates, ENF signal 65 with a qualifier of 40000000x is issued. AXR can be restarted by starting the AXRPSTRT procedure, which can be found in SYS1.PROCLIB. When the AXR address space initializes, an ENF signal of 80000000x is issued.

**Note:** See Chapter 15, “System REXX for z/OS” on page 237, for a detailed description of this new support.

## 19.2.2 SDSF REXX

SDSF is being enhanced to add the capability to provide access to SDSF functions through REXX variables. The variables are loaded with data from the SDSF panels. This enables them to be processed by REXX execs. The data can also be changed, which provides capabilities similar to those provided in the SDSF dialog by action characters and overtyping.

You can now access SDSF function with the REXX programming language. Using REXX with SDSF provides a simpler and more powerful alternative to using SDSF in batch.

**Note:** See Chapter 18, “SDSF enhancements” on page 291, for a detailed description of this new support.

## 19.2.3 Using REXX to write health check routines

IBM Health Checker for z/OS supports checks that are written in REXX using the new SYSREXX facility available with z/OS V1R9. This new SYSREXX facility makes it easier for you to write your own checks.

A REXX exec check consists of REXX language instructions that are interpreted and executed by System REXX. A REXX exec check runs in a System REXX address space in an environment defined by System REXX. IBM Health Checker for z/OS provides REXX functions as interfaces (HZSLSTART, HZSLFMSG, and HZSLSTOP) between a check and IBM Health Checker for z/OS and System REXX.

**Note:** See Chapter 22, “IBM Health Checker for z/OS” on page 365, for a detailed description of this new support.

## 19.2.4 XL C Metal compiler option

Before z/OS V1R9, all z/OS XL C compiler-generated code required Language Environment. In addition to depending on the C runtime library functions that are available only with Language Environment, the generated code depended on the establishment of an overall execution context, including the heap storage and dynamic storage areas. These dependencies prohibit you from using the XL C compiler to generate code that runs in an environment where Language Environment did not exist.

With z/OS V1R9, the XL C Metal compiler option generates code that does not have access to the Language Environment support at run time. Instead, the Metal option provides C-language extensions that allow you to specify assembly statements that call system services directly. Using these language extensions, you can provide almost any assembly macro, and your own function prologs and epilogs, to be embedded in the generated HLASM source file. When you understand how the Metal-generated code uses MVS(TM) linkage conventions to interact with HLASM code, you can use this capability to write freestanding programs.

**Note:** See Chapter 16, “z/OS XL C/C++ Metal option” on page 251, for a detailed description of this new support.

## 19.2.5 Common event adapter

Common event adapter (CEA) is a new component in the z/OS base with z/OS V1R9. Its has to be up and running in order for the CIM server to properly operate. It provides the ability to deliver z/OS events to C-language clients, such as the z/OS CIM server.

A CEA address space is started automatically during initialization of every z/OS V1R9 system. CEA has two modes of operation:

- ▶ Full function mode

In this mode, both internal z/OS components and clients (such as CIM providers) using the CEA application programming interface can use CEA functions.

- ▶ Minimum mode

In this mode, only internal z/OS components can use CEA functions.

The common event adapter (CEA) provides the ability to deliver z/OS events to C-language clients, such as the z/OS CIM server. The CEA address space is started automatically during z/OS initialization and does not terminate.

A set of new functions providing an “open face” through a client/server structure is implemented by the Common Information Model (CIM), also known as Pegasus. CIM instrumentations are z/OS extensions for providing the CIM server with information such as:

- ▶ z/OS cluster resource manager instrumentation
- ▶ Couple data sets instrumentation
- ▶ Jobs instrumentation

### Displaying the CEA environment

Use the **F CEA,DISPLAY** command to display information about the common event adapter (CEA) address space.

```
F CEA,DISPLAY
CEA0004I COMMON EVENT ADAPTER      734
STATUS: ACTIVE-FULL      CLIENTS: 0  INTERNAL: 0
EVENTS BY TYPE: #WTO: 0  #ENF: 0  #PGM: 0
```

The information displayed shows which activities are being monitored by CEA, and on behalf of which internal z/OS components and clients using the CEA application programming interface.

**Note:** It is possible in some circumstances to have CEA start in full function mode without this setup being performed. This is because of the way RACF could be configured to handle tasks that do not have user IDs of their own.

If CEA is running in minimum mode, you can change to full function mode by making these security definitions and then issuing the command **MODIFY CEA,MODE=FULL**.

## 19.3 Common Information Model

The Common Information Model (CIM) is a standard data model developed by a consortium of major hardware and software vendors (including IBM) known as the Distributed Management Task Force (DMTF) as part of the Web Based Enterprise Management (WBEM) initiative. CIM was introduced in z/OS with z/OS V1R7.



## Web Based Enterprise Management initiative

The Web Based Enterprise Management (WBEM) initiative includes a set of standards and technologies that provide management solutions for a distributed network environment. Interoperability is a major focus of WBEM, and using WBEM technologies can help you develop a single set of management applications for a diverse set of resources.

### 19.3.1 z/OS V1R9 enhancements for CIM

The following enhancements are introduced with z/OS V1R9 for CIM:

- ▶ **Automatic restart using the Automatic Restart Manager**

To extend the server availability, the CIM server is enabled to exploit the features of the Automatic Restart Manager (ARM). If an Automatic Restart Manager policy has been defined for CIM and the CIM server is authorized to register with ARM, then the CIM server will be automatically restarted by ARM.
- ▶ **New class IBMzOS\_LogicalDisk**

The base operating system instrumentation is extended to support logical disk volumes. A new class IBMzOS\_LogicalDisk is introduced, which inherits from CIM\_LogicalDisk and is associated to CIM\_ComputerSystem through the IBMzOS\_LogicalDiskDevice association.
- ▶ **New command line utility: cimsub**

A new CIM server command line utility lets you manage CIM indications on the local CIM server. This command can list, enable, disable, and remove indication subscriptions, filters and handlers.
- ▶ **Authentication based on SSL certificates**

The CIM server is enabled to exploit the AT-TLS facilities to authenticate CIM clients. AT-TLS provides a feature to enable and use SSL/TLS connections and encryption for communication with the CIM clients. Now the CIM server is aware of AT-TLS, and CIM clients can be authenticated through certificates.
- ▶ **CIM client for Java**

New with z/OS V1R9, the CIM Client for Java library from the SBLIM project is included with z/OS CIM. The CIM Client for Java is a programming API that enables z/OS applications written in Java for local and remote access of CIM instrumentation through the CIM-XML over HTTP access protocol. It consists of a Java library and associated online Java documentation.
- ▶ **Logging facility is changed to use the System Logger**

The CIM server logging facility is replaced by using the System Logger facility of the z/OS Communication Server. The logging to the z/OS system console is unchanged.
- ▶ **New instrumentation for job and sysplex management**

Instrumentation for server resources on the system are called *providers*. The providers, which are based on a subset of the standardized CIM classes, gather data on a system. CIM clients can work with these data by accessing the providers through the CIM server.

New instrumentation has been added for the management of jobs, sysplex, and Coupling Facility resources through CIM.
- ▶ **New and changed z/OS-specific messages**

Many new and changes messages are introduced with z/OS V1R9.

### 19.3.2 CIM cross-platform management

As shown in Figure 19-2, with CIM, management applications from different vendors can manage a heterogeneous environment of systems via the same technology. All applications operate on the same set of common data, such as the standard CIM Schema, using the same CIM-XML over HTTP access protocol. There is no need for vendors of management applications to either ship their own set of instrumentation nor to install their own agent technology on the systems to be managed.

Specific attributes of the various platforms are still available through CIM and can be either ignored or dynamically discovered by management applications. It is also still possible to create management applications for a specific platform only, by exploiting the extended CIM classes created for that platform. In providing this, it avoids the need for multiple management agents to be installed on the managed systems. The infrastructure for all types of management is the generic CIM Server.

As shown in Figure 19-2, a CIM client application requests the CIM server to return information about z/OS resources, which in this case is about basic z/OS data as well as RMF metrics. The CIM server invokes the appropriate CIM providers, which retrieve the requested data associated to z/OS system resources. The z/OS RMF monitoring provider invokes the RMF Distributed Data Server (DDS), which in turn collects RMF Monitor III performance data. The CIM server consolidates the data from the providers and returns them back to the calling client through the CIM/XML over HTTP access protocol.

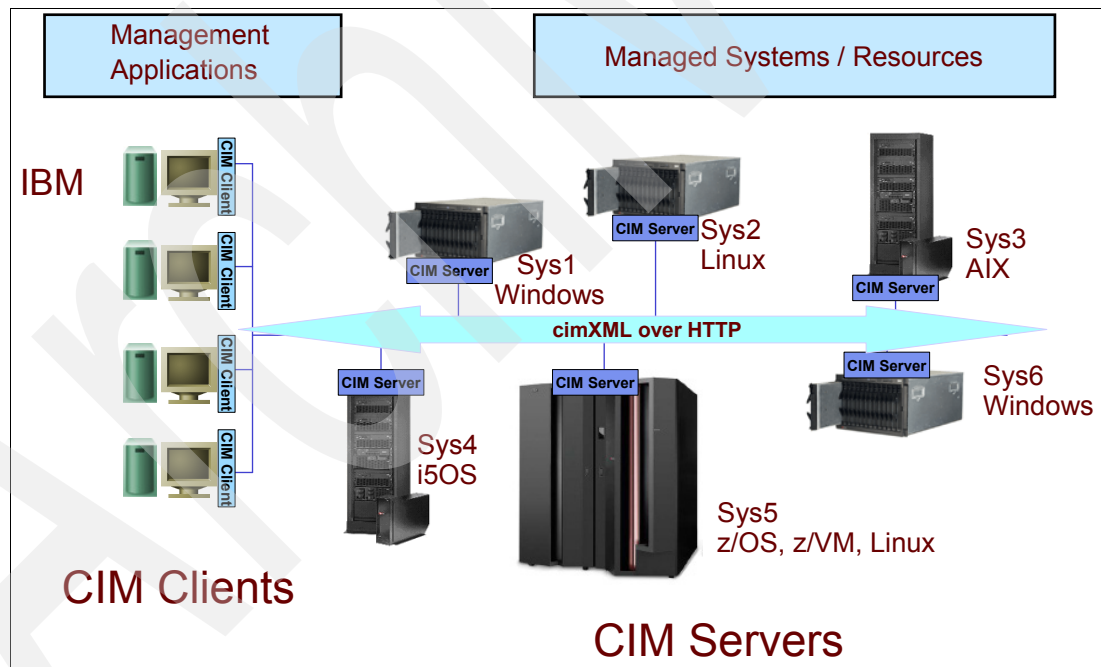


Figure 19-2 Cross-platform management

**Note:** The CIM-XML over HTTP protocol is an implementation of the standardized formats for communication between clients and the CIM server “Representation of CIM in XML” and “CIM Operations over HTTP”. For more information about these standards, see the WBEM Web site.

### 19.3.3 CIM components and dependencies

The overall goal of this open architecture is to provide simplified systems management functionality through a common, easy-to-use interface. However, you cannot just “slap” a GUI onto the operating system and expect something worthwhile to happen. The ability to interact with z/OS resources requires a concerted effort from the user interface (UI), to a remote interface, to a set of abstractions that organize the management operations functions provided by the operating system into understandable resource models, down to the low level functions that carry out these operations on z/OS itself.

With support for the CIM server on systems running z/OS, users have the ability to access z/OS resources through an extendible industry standard model. The CIM server, shown in Figure 19-3 on page 317, is used to receive client requests, collect the requested metrics/data from the managed system, and return the results to the client.

#### **CIM server support with z/OS V1R9**

Starting with z/OS 1R9, the CIM server exploits the functionality of common event adapter (CEA). CEA is a z/OS component that provides the ability to deliver z/OS events to C language clients. A CEA address space is started automatically during initialization of every z/OS system. In order for the address space to start successfully, you must configure CEA to work with z/OS. Failure to do so will cause CEA to run in a minimum function mode. For details, refer to the z/OS installation planning publication.

CIM provider access control permits the Communications Server CIM providers to gather CIM data when the user ID associated with the client of the z/OS CIM server is not defined as a superuser.

#### **CIM providers**

CIM data instrumentation is supplied by CIM components called providers. The providers gather data on a system in support of the CIM classes. Clients can retrieve the data through the Common Information Model Object Manager. On z/OS, this function is provided by the z/OS CIM server.

This CIM provider function resides in the /usr/lpp/tcpip/lib directory. There is no configuration necessary to activate this CIM provider support. The z/OS CIM server must be configured and activated for the data supported by the Communications Server CIM providers to be available to clients.

The z/OS Communications Server CIM classes are shipped with the z/OS CIM server. The files that define these classes and any platform-specific properties are also installed in the /usr/lpp/tcpip/mof directory.

Using CIM provides a consistent, modeled approach to providing system interfaces. The CIM server technology allows for clients to be running either on the same system or on other systems in the network.

The CIM providers describe and realize the model for each of the manageable resources on the system:

- ▶ Provide the XML interface for external callers
- ▶ Provide the interface code to enable an external caller (in this case, systems management applications) to obtain and alter state of various resources in the system

**Note:** IBM has developed providers for z/OS that support basic operating system information and some performance metrics. A CIM provider is the link between the CIM server and the system. This interface allows CIM to access and manage the resources. Each CIM provider makes accessible the resources it represents in a standard way.

### **Cluster instrumentation**

The cluster instrumentation work enables the cluster CIM provider to invoke the proper system interfaces to obtain and change the status of cluster-related resources:

- ▶ Internal services callable only by the Cluster Provider
- ▶ SYSREXX routines

In so doing, the providers extend the reach of sysplex systems management up to the CIM Server.

### **System REXX (SYSREXX)**

System REXX enables an authorized program to invoke a REXX script without having to go through the typical setup and management of either invoking the TSO TMP or having to manage REXX environments. Hence, using a simple programming interface, it may be run outside the normal TSO/E or batch environments.

It enables a low level program access to:

- ▶ Powerful parsing and string/character manipulation
- ▶ The ability to issue system commands and parse the results

It enables rapid development and deployment of system programmer tools, such as:

- ▶ Operations scripts
- ▶ Health checks

### **Common event adapter**

The z/OS common event adapter (CEA) enables a z/OS UNIX process to subscribe to various “legacy” asynchronous BCP events by type and matching criteria, as shown in Figure 19-4 on page 318, are as follows:

- ▶ WTO
- ▶ ENF
- ▶ Program-specified event

It extends the reach of “legacy” events to z/OS UNIX processes while components with existing events do not have to be reimplemented to communicate with CIM indication providers.

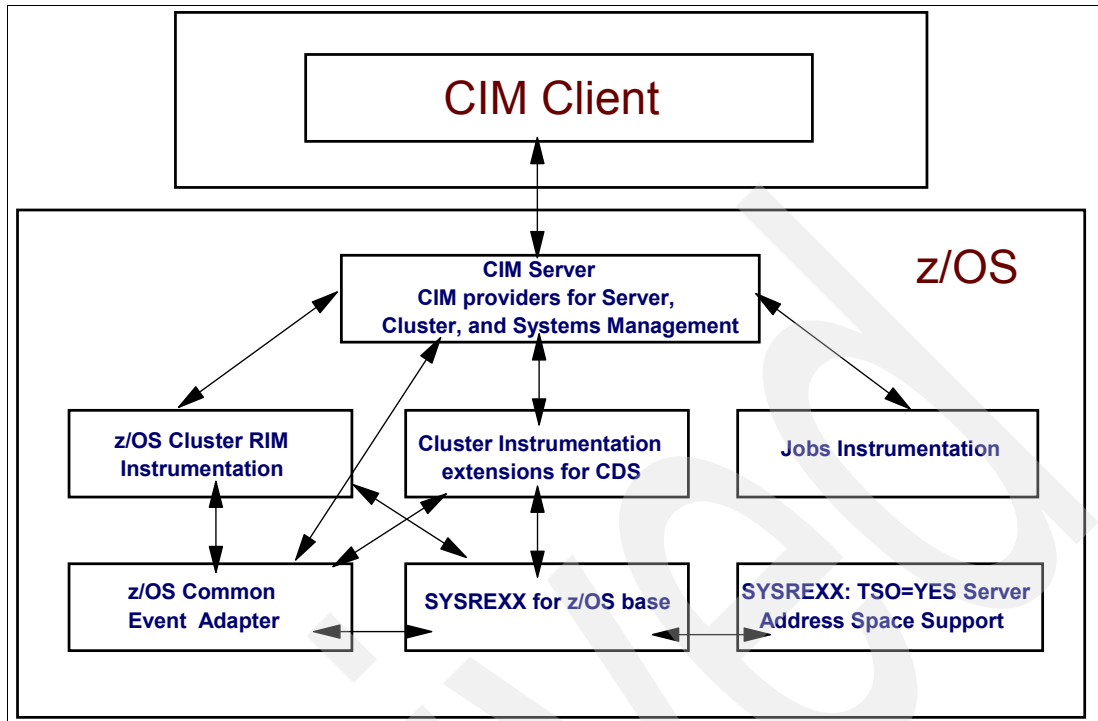


Figure 19-3 CIM Components and dependencies

### Jobs instrumentation

The jobs instrumentation enables the jobs and process CIM provider to invoke the proper system interfaces to obtain and change the status of batch jobs and z/OS UNIX processes; see Figure 19-4 on page 318.

The jobs instrumentation extends the reach of JES and z/OS UNIX Systems Services management up to the CIM server.

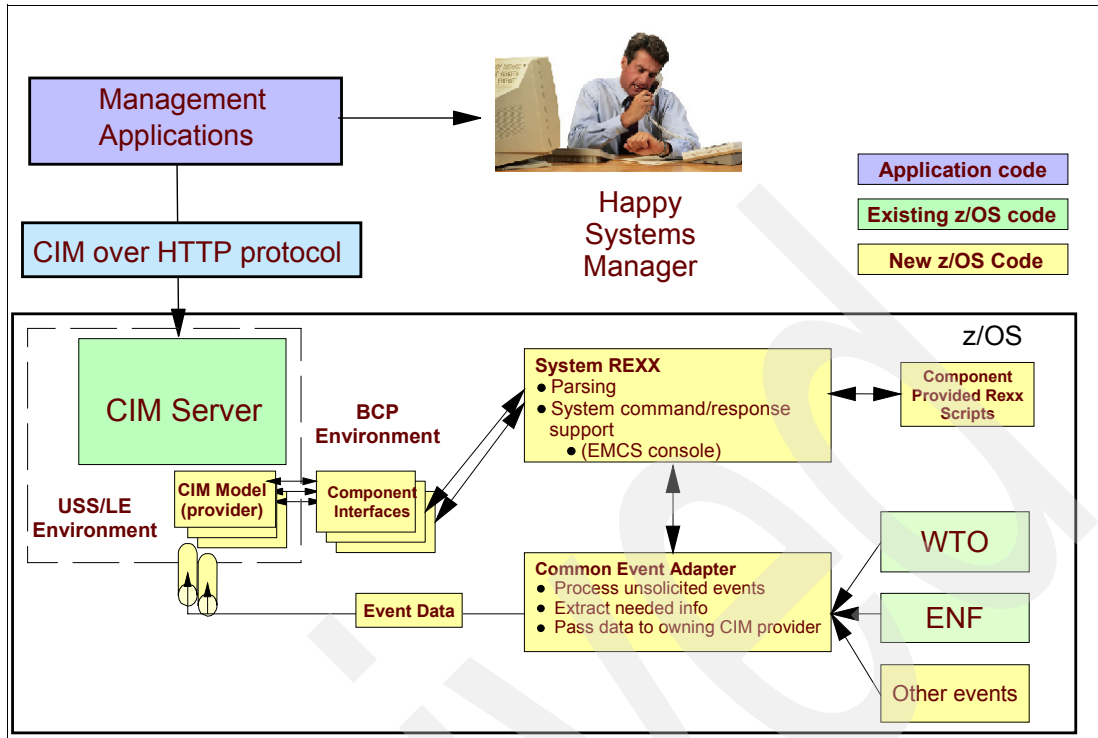


Figure 19-4 Cluster and jobs instrumentation

## 19.4 CIM server overview

For z/OS V1R7, a provider always ran in the same address space as the z/OS CIM server. This design had several undesirable side effects; for example, providers were able to crash the CIM server or other providers, intentionally or accidentally. To solve these problems, starting with z/OS V1R8, the Out-of-Process provider feature from OpenPegasus is ported to the z/OS CIM server and is enhanced to comply with the guidelines for z/OS security. The Out-of-Process feature manages providers in separate address spaces rather than loading and calling provider libraries directly within the CIM server process.

As mentioned, CIM data instrumentation is supplied by CIM components called providers. The providers gather data on a system in support of the CIM classes. Clients can retrieve the data through the Common Information Model Object Manager. On z/OS, this function is provided by the z/OS CIM server.

The CIM Object Manager (CIMOM), also known as a CIM server, is the software entity that receives, validates, and authenticates the CIM requests from the CIM client. It then directs the requests to the appropriate component or device provider.

### 19.4.1 CIM server support in z/OS V1R9

The CIM server for z/OS V1R9 has new components, as shown in Figure 19-5 on page 319.

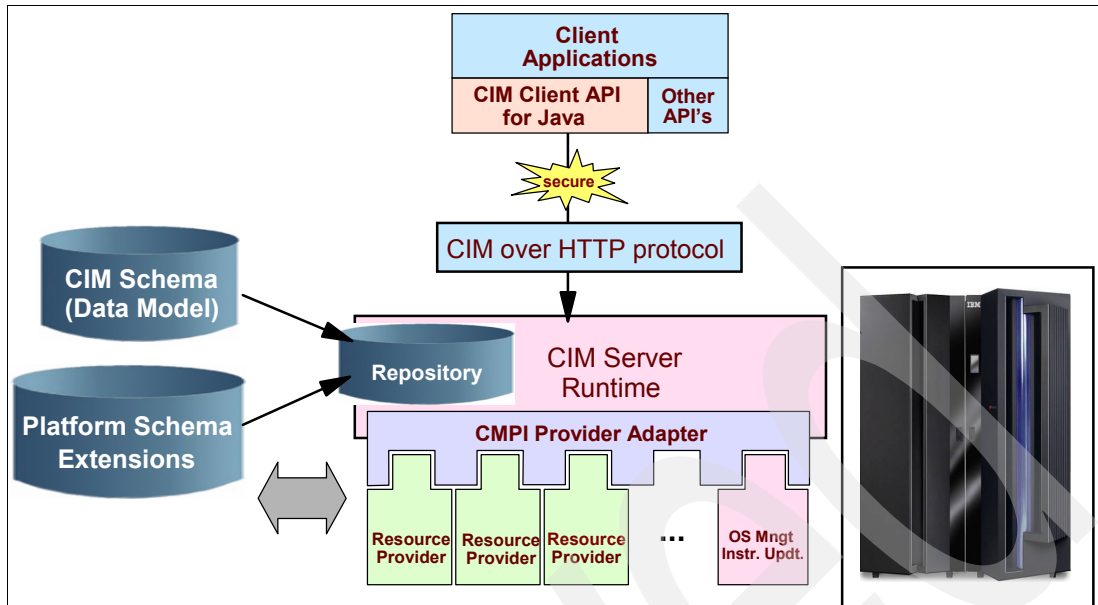


Figure 19-5 CIM server overview

These new components are described here:

► CIM client API for Java

The CIM client for Java library from the SBLIM project is included with z/OS CIM. The CIM Client for Java is a programming API that enables z/OS applications written in Java for local and remote access of CIM instrumentation through the CIM-XML over HTTP access protocol. It consists of a Java library and associated online Java documentation.

► CIM server runtime

The CIM server runtime environment security is split into authentication and authorization. Authentication is always enabled for the CIM server. The CIM server supports authorization via the RACF class WBEM, in which currently the single profile CIMSERV restricts access to the CIM server. With z/OS V1R9, this environment is enhanced with the following function support:

- General updates and improvements
- Automatic Restart Manager (ARM) support

To extend the server availability, the CIM server is enabled to exploit the features of the Automatic Restart Manager (ARM). If an Automatic Restart Manager policy has been defined for CIM and the CIM server is authorized to register with ARM, then the CIM server will be automatically restarted by ARM.

- SSL certificate-based authentication

The CIM server is enabled to exploit the AT-TLS facilities to authenticate CIM clients. AT-TLS provides a feature to enable and use SSL/TLS connections and encryption for communication with the CIM clients. Now the CIM server is aware of AT-TLS, and CIM clients can be authenticated through certificates.

- Logging facility is changed to use the syslog daemon
- New command-line utility: cimsub

The cimsub utility is a new CIM server command-line utility that allows you to manage CIM indications on the local CIM server. This command can list, enable, disable, and remove indication subscriptions, filters, and handlers.

- ▶ Operating system management instrumentation update

The CIM base classes is extended with a new class, IBMzOS\_LogicalDisk, which provides support for logical disk volumes. New instrumentation for job and sysplex management has been added for the management of jobs, sysplex, and Coupling Facility resources through CIM.

### **CIM security**

Because this support allows for the modification of key system resources, maintaining appropriate authorization is key, as follows:

1. The CIM server authenticates the end user.
2. The CIM server checks the authority of the user based on provider-level checks.
3. The credentials of the user are propagated from the CIM server, through the provider, down to the CEA, SYSREXX and other component interfaces, where authorization checks are made.

The secure check, shown in Figure 19-5 on page 319, requires the following security customization definitions when CEA is to be started in full function mode:

- ▶ Configure CEA to work with z/OS by updating the RACF database to permit CEA to use the Automatic Restart Manager (ARM). Use this command:

```
ADDUSER CEA DFLTGRP(SYS1) OMVS(UID(0) HOME('/') FILEPROCMAX(1024)) SPECIAL
RDEFINE STARTED CEA.** STDATA(USER(CEA) GROUP(SYS1) TRACE)
```

- ▶ Define the OMVS segment that allows CEA to work in the UNIX environment. Use this command:

```
ADDUSER userid DFLTGRP(SYS1) OMVS(UID(0) HOME('/') FILEPROCMAX(1024))
SPECIAL
```

The RACF user ID *may* be CEA but, because CEA is a started task, it does not *have* to be. If the STARTED class or started procedures table (ICHRIN03) contains another user ID, CEA will have that user ID assigned to it.

For example, a generic entry might specify that a user ID such as STCUSER or IBMUSER should be assigned to any started task that is not defined with its own entry. If the user ID CEA was not set up and assigned to the started task, then the generic entry would be used and an IEF695I message would indicate that START CEA was assigned to the generic user ID.

If the default user ID that is assigned does not have an OMVS segment, a default OMVS segment is sought through the FACILITY class profile BPX.DEFAULT.USER. RACF does not use the default OMVS segment unless the task is running with a RACF-defined user ID.

### **CMPI provider adapter**

CMPI is a C-based programming interface for providers designed for binary compatibility. All management instrumentation included with the z/OS CIM server was developed following the CMPI standard, and CMPI is the only supported provider programming interface for the z/OS CIM server. Documentation about the CMPI Technical Standard is available from The Open Group and is not repeated in any documentation available for z/OS.

Developers of management instrumentation for z/OS need to be familiar with the CMPI and CIM/WBEM standards. The information contained here explains the specific aspects that need to be considered for developing CMPI.



Management instrumentation is implemented by developing a provider. A provider is a dynamic load library (DLL) that implements a given interface and contains the program code used by the CIM server to interact with the system resource described by a certain CIM class, for example CIM\_Processor.

Providers are registered with the CIM server for a defined CIM class, allowing the CIM server to route all client requests directed against this class to the provider for interacting with the resource. A provider logically acts as an extension of the CIM server for interfacing directly with the managed resources.

There are also a few samples for CMPI providers available on the OpenPegasus CVS Repository. They can be obtained the same way as the header files, by navigating to the `pegasus/src/Providers/sample/CMPI` directory.

**Important:** Users familiar with CVS can check out these files using a CVS client on any platform by following the instructions in the CVS Overview section at

<http://www.openpegasus.org/>

The required files are located in directory `pegasus/src/Pegasus/Provider/CMPI`. To get the correct version of the files, they need to be checked out with the `RELEASE_2_5_1` tag.

Users who are not familiar with using CVS should obtain the files through a Web browser starting at:

<http://www.openpegasus.org/>

From the main page switch to the Web CVS section, where you can navigate to the required CMPI files by clicking the following directory names:

pegasus (see Figure 17) > src > Pegasus > Provider > CMPI

**Note:** Before you can start to develop a CMPI provider, you first need to have the CIM class model containing descriptions for the resource to be instrumented in the form of a CIM class. Such a class should follow the standards from DMTF Standards and Initiatives, and in particular it should be consistent with the CIM Schema supported by the CIM server.

Usually, a CIM class for which a provider is written is derived from one of the classes in the CIM Schema provided by the DMTF, and named with a vendor-specific class name prefix. For example, the prefix `IBMzOS_` is used for all classes provided by IBM for the z/OS operating system.

This naming scheme also helps to prevent conflicts with the resources that have already been instrumented for CIM by IBM or other vendors. In general it is not recommended to create new providers for resources that have already been instrumented by IBM.

The DMTF CIM Schema defines an information model for representing systems management functions. For z/OS V1R9, CIM Schema version 2.11 is supported by the CIM server.

## 19.5 CIM client-to-CIM server access

Communication between the CIM server and a CIM client using AT-TLS and RACF uses a client's user ID and a password. Starting with release the z/OS V1R9 CIM server, you can

use SSL certificates and encryption for communication with the CIM clients, because AT-TLS is enabled to authenticate CIM clients by SSL certificates in cooperation with RACF, as shown in Figure 19-6. The CIM client sends an SSL certificate to AT-TLS, AT-TLS sends the certificate to RACF, and RACF associates the certificate to the appropriate user ID, which then can access the CIM server. Vice versa, the CIM server returns its responses to the client's requests using SSL certificates.

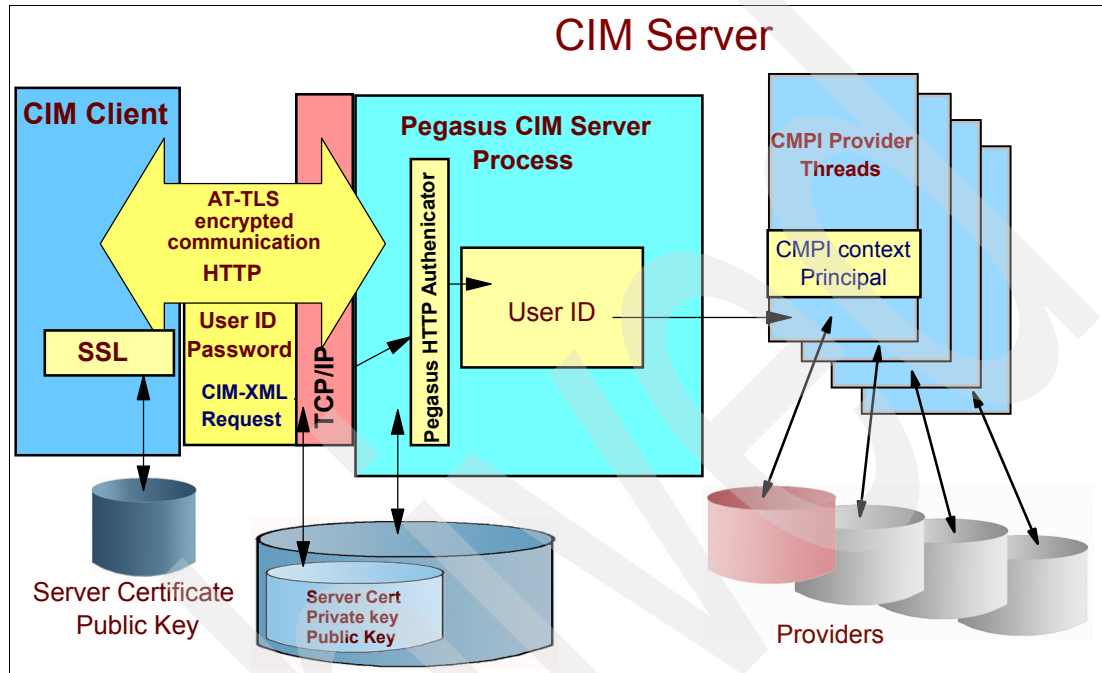


Figure 19-6 CIM client logging on to a CIM server

## CIM server

For the CIM server for z/OS, users log in over HTTP or HTTPS using basic authentication. When logging in, users are authenticated using their z/OS user ID and password as defined, for example, in the Resource Access Control Facility (RACF).

A CIM user must have at least READ access to the CIMSERV RACF profile in the WBEM class to access the CIM server. In order to use any of the administrative command line tools of the CIM server, a user instead requires CONTROL access to the CIMSERV profile.

The CIM server authenticates users with the z/OS Security Server RACF to determine which users can log into it. Authentication is performed for every new connection (local or remote) before a user is granted access to the CIM server, as shown in Figure 19-6.

The CIM server offers an optional authorization check. This check is optionally performed on a per provider basis, meaning that a RACF profile in the WBEM class can be related to a single provider library. Correlation between a provider and a RACF profile occurs during provider registration by the addition of a property in the PG\_Provider class.

The provider-based authorization is defined by the vendor of a provider rather than by the CIM server administrator. Therefore, specific RACF requirements will need to be documented on a per provider basis.

## 19.6 CIM server runtime update and enhancements

The objective of the CIM updates for z/OS V1R9 is to keep the CIM server for z/OS up to date and at same level with other e-Server platforms.

It consists of the following enhancements:

- ▶ TCP/IP restart toleration

This allows TCP/IP to be stopped and started. Requests currently running are terminated. A single error message is issued:

```
CFZ09100I: TCP/IP temporary unavailable
```

After restarting TCP/IP, all sockets are reconnected again without operator invention.

- ▶ CIM error support

This is an extension to incorporate the DMTF-defined functionality to allow CIM\_Error instances to be created by the server and passed to the client as components of CIM/XML.

- ▶ Support for embedded instances

This support aims to integrate embedded instances into the Pegasus infrastructure, handling the encoding, parsing, and manipulation of embedded instance properties and parameters in the same manner as numerics, strings, embedded and objects.

The following topics are enhancements to the CIM server in order to receive benefit of z/OS qualities of service.

- ▶ ARM support
- ▶ SSL certificate authentication
- ▶ CIM logging to syslog daemon (syslogd)
- ▶ cimsub command

### 19.6.1 Automatic Restart Manager support

The CIM server (only if started as a started task) automatically registers with Automatic Restart Manager (ARM), in which case it issues the following successful registration message:

```
CFZ12532I: The CIM server successfully registered to ARM using element name  
CFZ_SRV_SC63
```

The CIM server will be de-registered from ARM during the normal shutdown procedure. In case of failure, the following message is issued and if you do not plan to use ARM, ignore the warning message:

```
CFZ12533W: The CIM server failed to register with ARM using element name  
CFZ_SRV_SC63 : return code 0x08, reason code 0x0134.
```

#### **ARM element name**

The ARM element name CFZ\_SRV\_<SYSNAME> has to be defined as shown in Figure 19-7 on page 324.

```

DEFINE POLICY NAME(CFZARMP0) REPLACE(YES)

  RESTART_GROUP(CFZCIMRESGRP)
  /* List all systems where the CIM Server can be started */
  TARGET_SYSTEM(SC63,SC70)
  /* Wait 10 sec before restarting to free resources */
  RESTART_PACING(10)

  ELEMENT(CFZ_SRV_*)
  RESTART_ATTEMPTS(3,300)
  RESTART_TIMEOUT(300)
  READY_TIMEOUT(300)
  /* cross-system restart is not allowed. */
  /* No restart after system failure */

  TERMTYPE(ELEMTERM)
  RESTART_METHOD(ELEMTERM,STC,'S CFZCIM')

```

Figure 19-7 Sample ARM policy

In a sysplex, no cross-system restarts are allowed, because there is always one CIM server per MVS image.

### SAF configuration for ARM

For security reasons, the IXCARM.DEFAULT.CFZ\_SRV\_\* profile has to be defined in the FACILITY class of RACF and the CIM server UID must have UPDATE access to that profile, as shown here:

```

RDEFINE FACILITY IXCARM.DEFAULT.CFZ_SRV_* UACC(NONE)
PERMIT IXCARM.DEFAULT.CFZ_SRV_* CLASS(FACILITY) +
      ID(CFZADM) ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST (FACILITY)
SETROPTS RACLIST(FACILITY) REFRESH

```

## 19.6.2 SSL certificate-based authentication

The CIM server for z/OS V1R9 provides an additional authentication mechanism using certificates, as shown in Figure 19-8 on page 325. The CIM server is using the z/OS Communications Server Application Transparent Transport Layer (AT-TLS) and more specifically it is using its Policy Agent. The CIM server is AT-TLS-aware, and it queries AT-TLS for the z/OS User ID.

Certificate management must be performed by an SAF-compliant product (for example RACF), which does the following:

- ▶ Stores CA and CIM server certificates into key-rings
- ▶ Maps certificate subjects to a z/OS user ID, either one-to-one or many-to-one

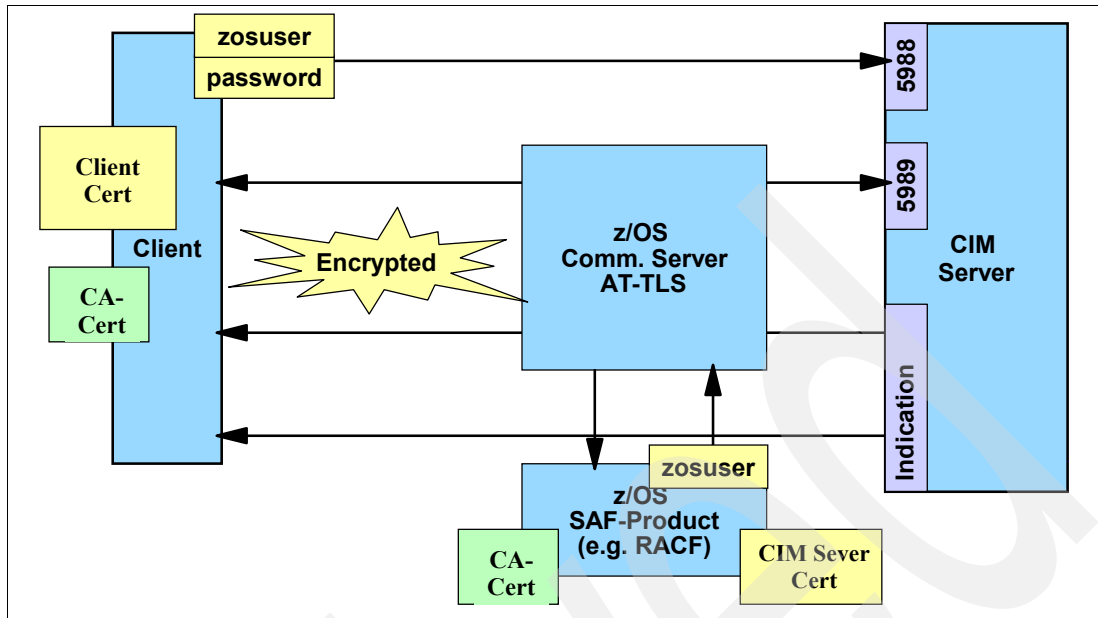


Figure 19-8 SSL certificate-based authentication

## SSL certificate-based authentication configuration

The SSL certificate-based authentication configuration is executed through the `cimconfig` command, as follows:

```
-enableHttpsConnection=true
-httpsPort=5989
```

## The AT-TLS configuration

The definitions are performed through a Policy Agent rule, which is part of z/OS Communications Server configuration, as follows:

```
“Jobname” or “Username” for policy selection.
“LocalPortRange” must match httpsPort of CIM Server
“HandshakeRole” set to ServerWithClientAuth
“ClientAuthType” set to SAFCheck
“TTLSTKeyringParms” set to the SAF managed keyring
“ApplicationControlled” set to OFF
```

When using SSL client authentication with the z/OS CIM server, in AT-TLS the parameter `HandshakeRole` has to be set to `ServerWithClientAuth`. Otherwise, the `HandshakeRole` has to be set to `Server` on the inbound AT-TLS policy configuration.

The z/OS CIM server requires the `HandshakeRole=ServerWithClientAuth` or `HandshakeRole=Server` AT-TLS policy option to be set. If not set, the connection will be rejected.

Note that the AT-TLS policy values `Full` and `Required` for the property `ClientAuthType` are *not* recommended settings for the z/OS CIM Server, because an additional HTTP Basic authentication (user ID/password) is then required.

The CIM client is verified according to one of the following rules shown in Figure 19-9 on page 326.


AT-TLS Policy parameter ClientAuthType	z/OS CIM Server behavior
PassThru	The client is not prompted for a certificate. The authentication is done through HTTP Basic authentication (user ID/password).
Full <b>(Not Recommended)</b>	If the client provides a certificate, it is validated. Whether or not it is valid, HTTP Basic authentication is done. <b>User ID/password still required!</b>
Required <b>(Not Recommended)</b>	The client has to provide a valid certificate. The CIM Server only does HTTP Basic authentication if it is valid. <b>User ID/password still required!</b>
SAFCheck 	The client has to provide a valid certificate and this certificate is associated with a user ID. No HTTP Basic authentication is done.

Figure 19-9 Client verification mode

### 19.6.3 Logging facility changed to syslog daemon

In CIM with z/OS V1R9, files \*.log within the /var/wbem/logs directory are no longer used. Instead, the syslog daemon (syslogd), part of the z/OS Communication Server, is used. If this daemon is not yet active, refer to the following publications for detailed explanations about how to start it:

- ▶ *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- ▶ *z/OS Communication Server: IP Configuration Guide*, SC31-8775

The logging utility that is available for the CIM server, and which is interfacing with syslog, is initially enabled and cannot be disabled. However, you can configure the utility by choosing the log level that you want to use: TRACE, INFORMATION, SEVERE or FATAL.

TRACE returns the most detailed trace information, while FATAL only produces trace output for fatal errors.

You can change the logging level by using the logLevel parameter of the **cimconfig** command while the CIM server is running. For example, you can go into the USS shell and type the following command while the CIM server is running:

```
cimconfig -s logLevel=INFORMATION
```

### 19.6.4 New command-line utility: cimsub

The **cimsub** command lets you manage CIM indications on the local CIM server. The command can list, enable, disable and remove indication subscriptions, filters and handlers. However, you cannot modify or create a handler or a filter. The CIM indication must be created or modified by a CIM client program.

In order to use the **cimsub** command, the CIM server must be running on the local system and a user needs to have CONTROL access to profile CIMSERV in WBEM class.

The following example lists all subscriptions in the namespace root/PG\_InterOP in verbose mode when issuing the following command and receiving the following messages:

```
cimsub -ls -v  
Namespace: root/PG_InterOp Filter:  
root/PG_InterOp:IndicationTest_indicationFilter  
Handler: root/PG_InterOp:CIM_ListenerDestinationCIMXML.IndicationTest  
Query: "SELECT * FROM TestIndication"  
Destination: http://test.server.com/  
SubscriptionState: Enabled
```

To disable the subscription specified by -F and -H, and display the result in verbose mode, issue the following command and receive the following messages:

```
cimsub -d -F IndicationTest_indicationFilter -H IndicationTest cimsub -ls -v  
Namespace: root/PG_InterOp  
Filter: root/PG_InterOp:IndicationTest_indicationFilter  
Handler: root/PG_InterOp:CIM_ListenerDestinationCIMXML.IndicationTest  
Query: "SELECT * FROM TestIndication"  
Destination: http://test.server.com/  
SubscriptionState: Disabled
```

## 19.7 CIM client API for Java

A CIM client API for Java is provided to address the requirement of a Java-based client API to exploit the CIM server on z/OS; see Figure 19-10 on page 328. This is expected to enable vendors and other exploiters to write CIM client applications in Java on z/OS.

It supports the CIM-XML over HTTP(S) protocol. The interface, which is not yet standardized, uses JSR. The code is located at /usr/lpp/wbem/jclient. Version 1.3 is shipped under the form of a client library named sblimCIMClient.jar, with a configuration file located in cim.defaults. API Java documentation is available in sblim-cim-client-doc.zip. Further information can be found at:

<http://sourceforge.net/projects/sblim/>

The SBLIM CIM client is a pure Java-based implementation of the following:

- ▶ The WBEM operations API
- ▶ The CIM metamodel representation
- ▶ An indication listener

In CIM terminology, an *indication* is the representation of the occurrence of an event. For example, an event can be the unexpected termination of a program, or the modification of a property value of a CIM instance. There is not necessarily a one-to-one correspondence between events and indications. In particular, multiple indications can be generated for the same underlying event if multiple CIM client applications had subscribed for the event.

An event can also occur without causing a related indication to be raised (for example, if no subscription was made for the event).

**Note:** Additional examples are available from the SBLIM OpenSource project and can be downloaded from:

<http://www.sblim.org>

Although the CIM providers from SBLIM apply to Linux platforms only, they are useful examples of how to write CIM providers in general. The SBLIM project also provides a number of useful tools and documents related to provider development.

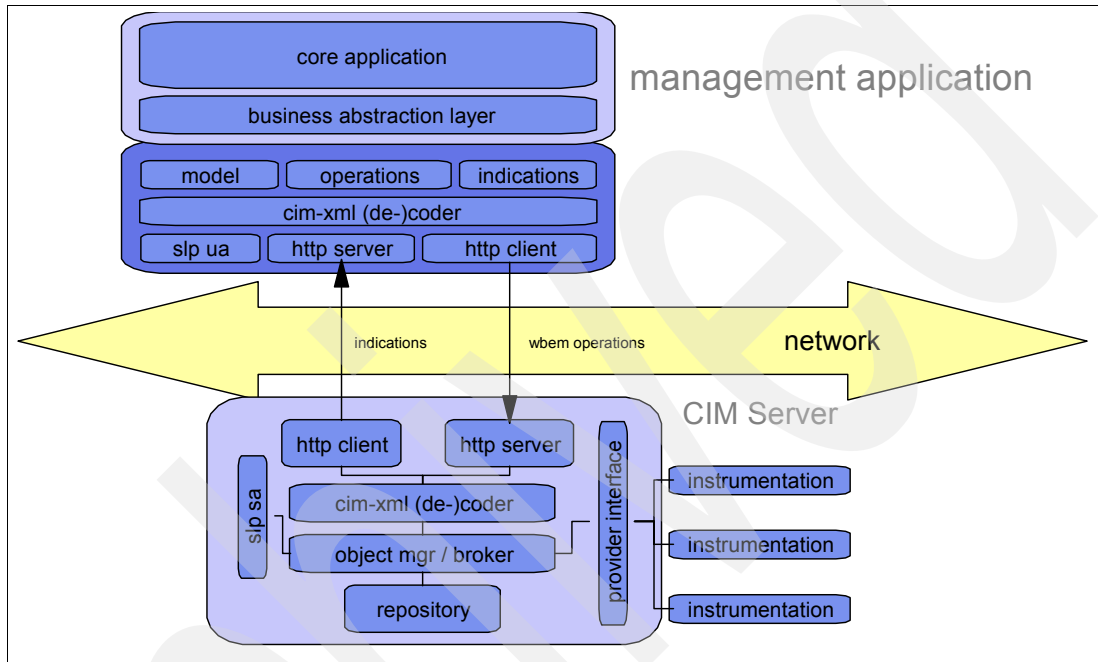


Figure 19-10 CIM client API for Java

## 19.8 Instrumentation in z/OS V1R9

The CIM standard provides the ability to develop management applications that work with systems management data. To work with CIM, developers should have a thorough understanding of the CIM standard defined by the DMTF.

**Note:** For more information about the CIM standard, see Common Information Model (CIM) Standards on the DMTF web site. For information about z/OS CIM, see *z/OS Common Information Model User's Guide*, SC33-7998.

IBM has developed providers for z/OS that support basic operating system information and some performance metrics. A CIM *provider* is the link between the CIM server and the system. This interface allows CIM to access and manage the resources. Each CIM provider makes accessible the resources it represents in a standard way.

The CIM Server provides interfaces that allow CIM clients to get and set management information on the system. The management information is based on the CIM model of managed objects and is provided to clients in CIM-HTML format. The models contain schemas of classes used to represent managed objects, and the associations between the classes.



The classes are static descriptions of managed object types. Dynamic data about the state of physical managed objects is obtained from CIM providers, and is used to create instances of the classes. CIM instrumentation in z/OS V1R9 has been enhanced by a certain number of new CIM classes.

A part of z/OS CIM is the eServer OS management profile. It provides access to, and management of, the base IT resources, as follows:

- ▶ zSeries LPARs
- ▶ z/OS operating system images
- ▶ z/OS address spaces

### **CIM client/server implementation**

The z/OS base element Common Information Model (z/OS CIM) implements the CIM server, based on the OpenPegasus open source project. A CIM monitoring client invokes the CIM server, which in turn collects z/OS metrics from the system and returns it to the calling client.

Figure 19-11 on page 330 illustrates how the CIM server works in the z/OS environment, as follows:

1. A CIM client application requests the CIM server to return information about z/OS resources, in this case about basic operating system (OS) data as well as RMF metrics.
2. The CIM server invokes the appropriate CIM providers, which retrieve the requested data associated to z/OS system resources.
3. The z/OS RMF monitoring provider invokes the RMF Distributed Data Server (DDS), which in turn collects RMF Monitor III performance data.
4. The CIM server consolidates the data from the providers and returns them back to the calling client through the CIM-XML over HTTP protocol.

In addition, Figure 19-11 on page 330 shows two types of CIM providers:

- ▶ RMF monitoring providers that use the RMF DDS to access the z/OS system data  
A CIM monitoring client invokes the CIM server which, in turn, collects z/OS metrics from the system and returns it to the calling client. To get the z/OS metrics, the CIM server invokes the z/OS RMF monitoring provider, which retrieves the metrics associated with z/OS system resources. The z/OS RMF monitoring provider uses existing and extended RMF Monitor III performance data. The metrics obtained by this new API are common across eServer platforms, so you can use it to create end-to-end monitoring applications.
- ▶ z/OS operating system management providers that access the z/OS system data directly

### **IBM-supplied providers**

CIM classes have been implemented as IBM-supplied providers, and this is implemented according to the DMTF dynamic metrics model. You can find more information about this data model in the CIM Metrics white paper (DSP0141), which is available at the DMTF Web page in the CIM White Paper section at the following URL:

<http://www.dmtf.org>

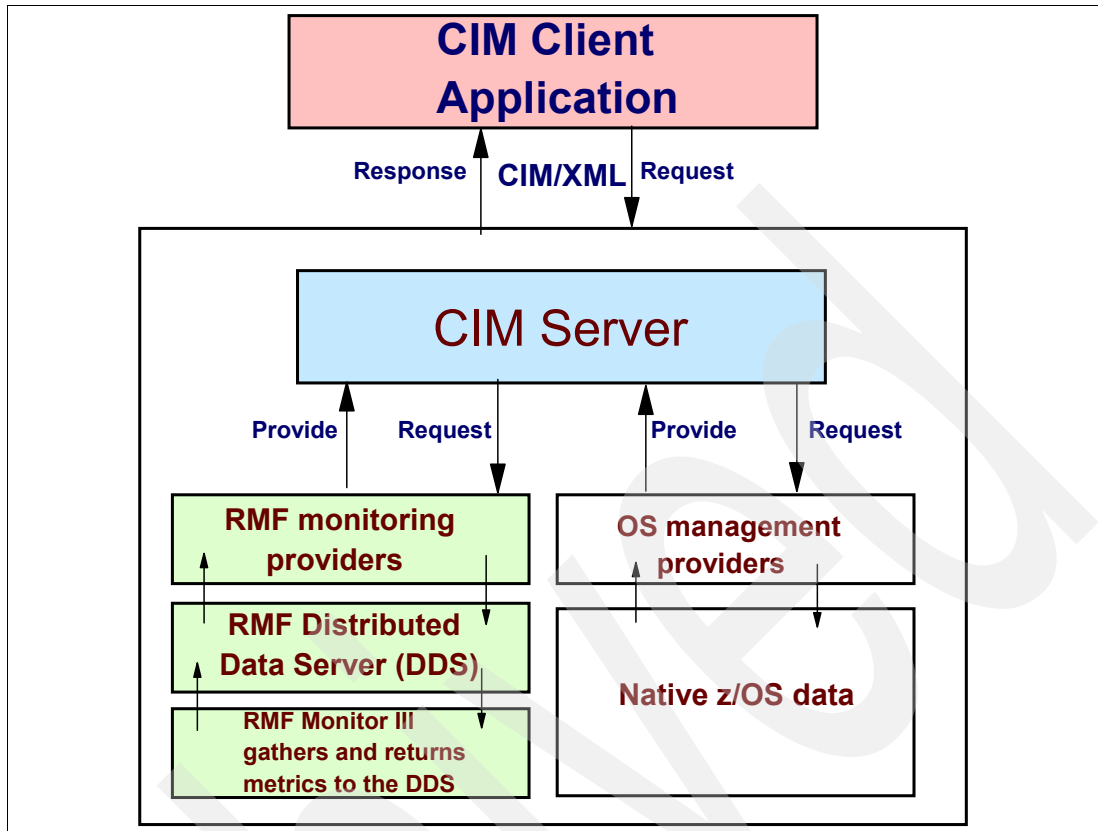


Figure 19-11 Client/Server requests for data to z/OS using CIM

### 19.8.1 Required parmlib updates

The following parmlib parameters have to be defined to enable the job and cluster providers:

- ▶ MAXCAD limit

This parameter defaults to 50. If the installation sets a lower limit, it may be necessary to increase this setting to accommodate the Common Event Adapter (CEA) Common Area Data Space (CADS).

- ▶ APF authorize SYS1.MIGLIB

The following must be added to the installation's PROGxx member in PARMLIB to enable the CFRM-related CIM providers to function:

```
APF ADD DSNAME(SYS1.MIGLIB) VOLUME(*****)
```

- ▶ REXX alternate library

The couple data set providers require the use of compiled REXX execs provided as part of the z/OS V1R9 SYSREXX support. These execs require the use of the REXX alternate library. The following addition to the installation's PROGxx member in PARMLIB is one way to accomplish this:

```
LNLKST ADD,NAME(LNLKST00),DSN(REXX.V1R3M0.SEAGALT),ATTOP
```

**Note:** A sample TSO CLIST is provided that performs the necessary RACF setup to permit CEA to use Automatic Restart Manager (ARM), and to permit CEA to operate in UNIX System Services (USS) with the cluster, couple data set, and JES2/JES3 jobs CIM providers.

## 19.8.2 Instrumentation for logical disk volumes

The base operating system instrumentation is extended to support logical disk volumes. A new IBMzOS\_LogicalDisk class is introduced, which inherits from CIM\_LogicalDisk and is associated to CIM\_ComputerSystem through the IBMzOS\_LogicalDiskDevice association.

For basic operating system management data, a set of CIM classes is provided, such as the logical disk volumes, the properties of which are as shown in Figure 19-12.

Property Name	Schema Class	Comments
string Caption	CIM_ME	Always returns "z/OS Storage Volume"
string Description	CIM_ME	Always returns "Represents a storage volume as seen by z/OS"
string ElementName	CIM_ME	Same as Name (Volume Serial Number)
string Name	CIM_MSE	Volume Serial Number obtained from UCBVOLI.
uint16 OperationalStatus	CIM_MSE	[0] = "Unknown"
uint16 EnabledState	CIM_ELE	Mapped from the UCBONLI and UCBBOX values. (See next chart for mapping)
string CreationClassName [key]	CIM_LDev	Always returns "IBMzOS_LogicalDisk"
string DeviceID [key]	CIM_LDev	Channel Device ID obtained from UCBCHAN.
String[] IdentifyingDescriptions	CIM_LDev	[0]="Device Node Element Descriptor"
String[] OtherIdentifyingInfo	CIM_LDev	[0]=Device Node Element Description obtained from the NEDID field of the matching IHACDR control block.
string SystemCreationClassName [key]	CIM_LDev	Always returns "IBMzOS_ComputerSystem"
string SystemName [key]	CIM_LDev	The systems fully qualified hostname (see IBMzOS_ComputerSystem::Name)

Figure 19-12 IBMzOS\_LogicalDisk – implemented properties

CIM instrumentation added for logical disk allows monitoring of logical disk volumes through open standards. This enhancement is designed for the heterogeneous management of operating systems.

## 19.8.3 Instrumentation of batch jobs

With z/OS V1R9, IBM z/OS instrumentation of batch jobs allows the CIM provider to obtain generic information about a z/OS jobs, such as its jobname, priority, JES properties, and status (Cancel, Hold, Release, Restart).

It also allows the CIM provider to obtain generic information about the name of the subsystem and its characteristics (JES2 or JES3, primary subsystem or not), sysout data sets and any details of this kind.

IBMzOS\_Job is a new class that represents a z/OS job. Jobs are associated with a subsystem, such as JES2, JES3, or MSTR. Some properties may require significant overhead, including I/O, to obtain their data. These properties are identified with the qualifier of Expensive. To reduce system overhead, the provider will only return the values for these expensive properties if they are explicitly requested by name.

## 19.8.4 Instrumentation for a sysplex

A large set of CIM instrumentation regarding a sysplex is added in z/OS V1R9, as shown in Figure 19-13 on page 332.

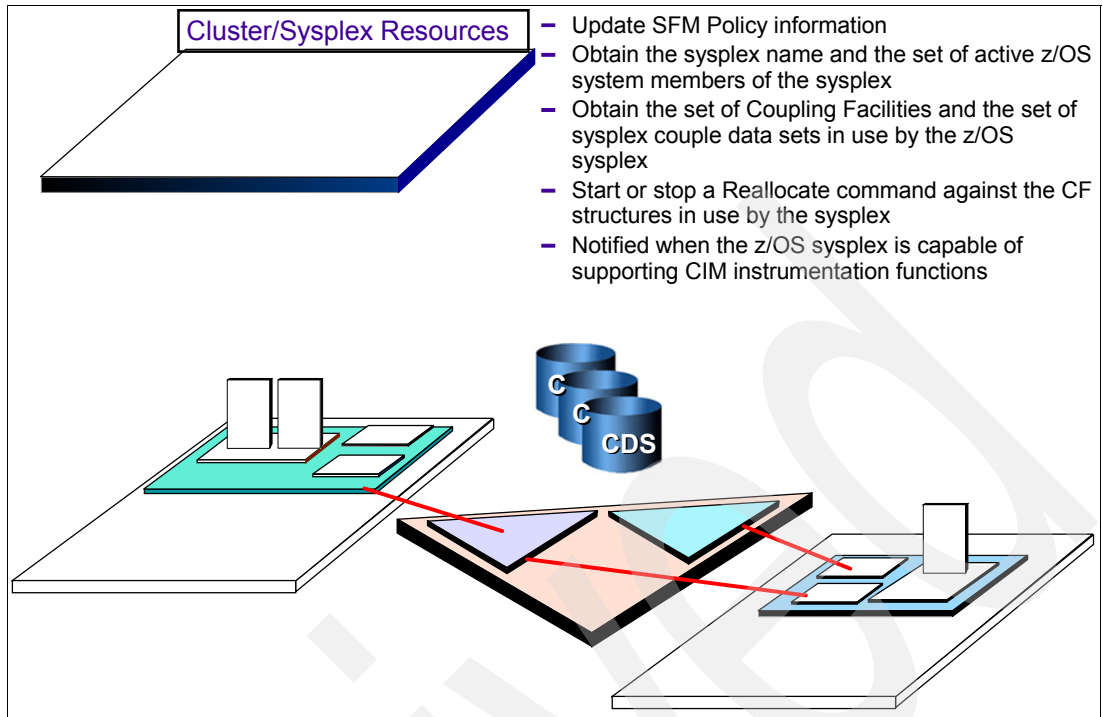


Figure 19-13 CIM Instrumentation: Cluster/Sysplex resources

Each sysplex node (that is, LPAR) is instrumented as shown in Figure 19-14.

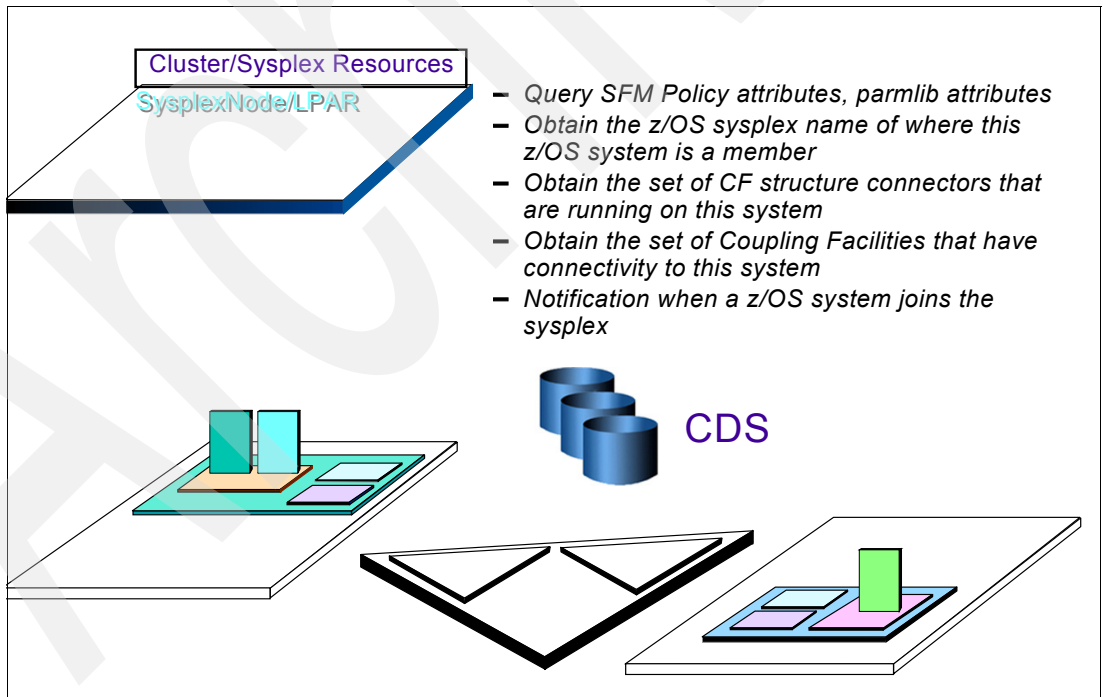


Figure 19-14 CIM Instrumentation: SysplexNode/LPAR

The Coupling Facility is itself instrumented as shown in Figure 19-15 on page 333.

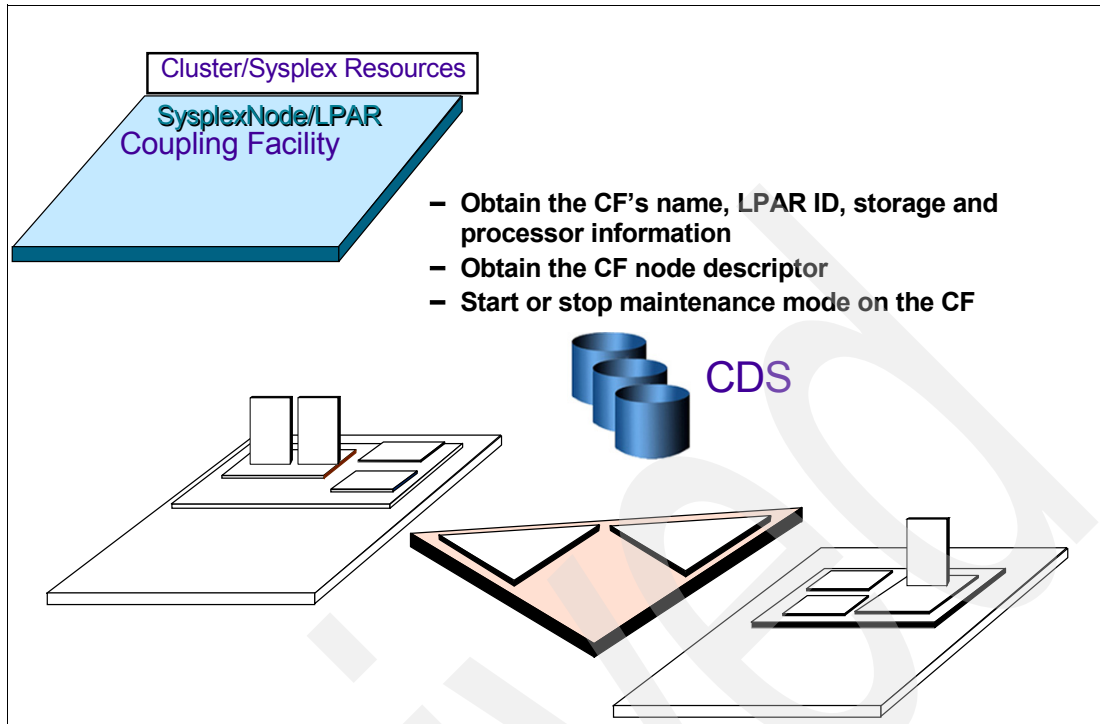


Figure 19-15 CIM instrumentation: Coupling Facility

Each Coupling Facility structure is instrumented as depicted in Figure 19-16.

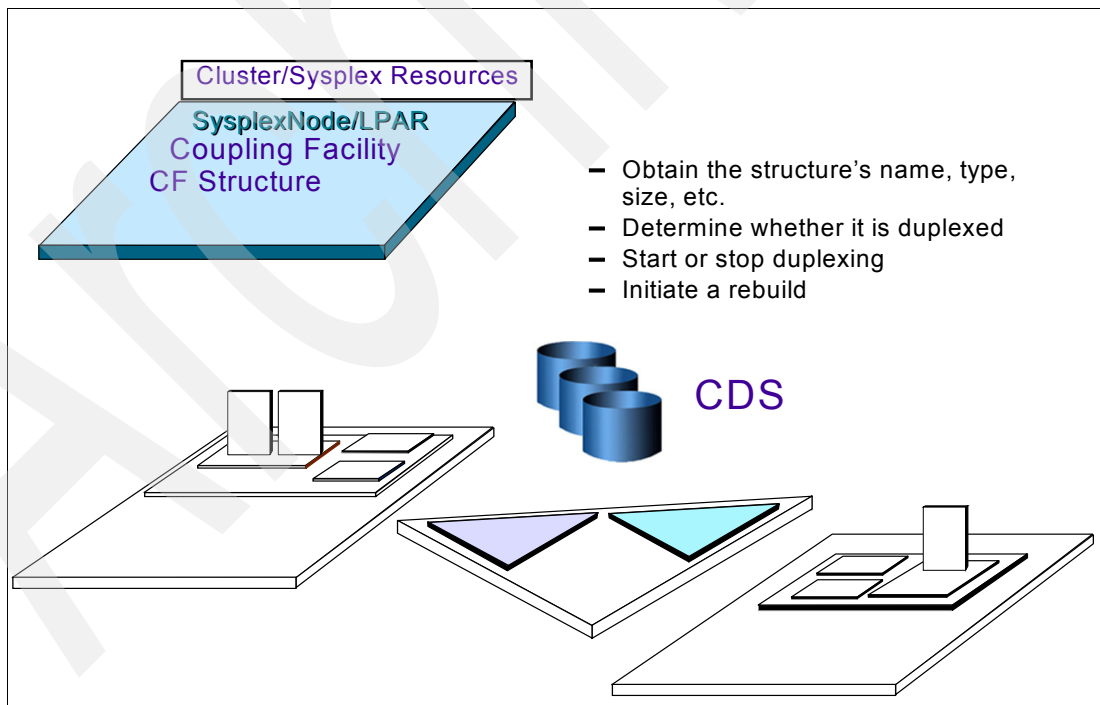


Figure 19-16 CIM instrumentation: CF structure

CF connectors are instrumented as depicted in Figure 19-17.

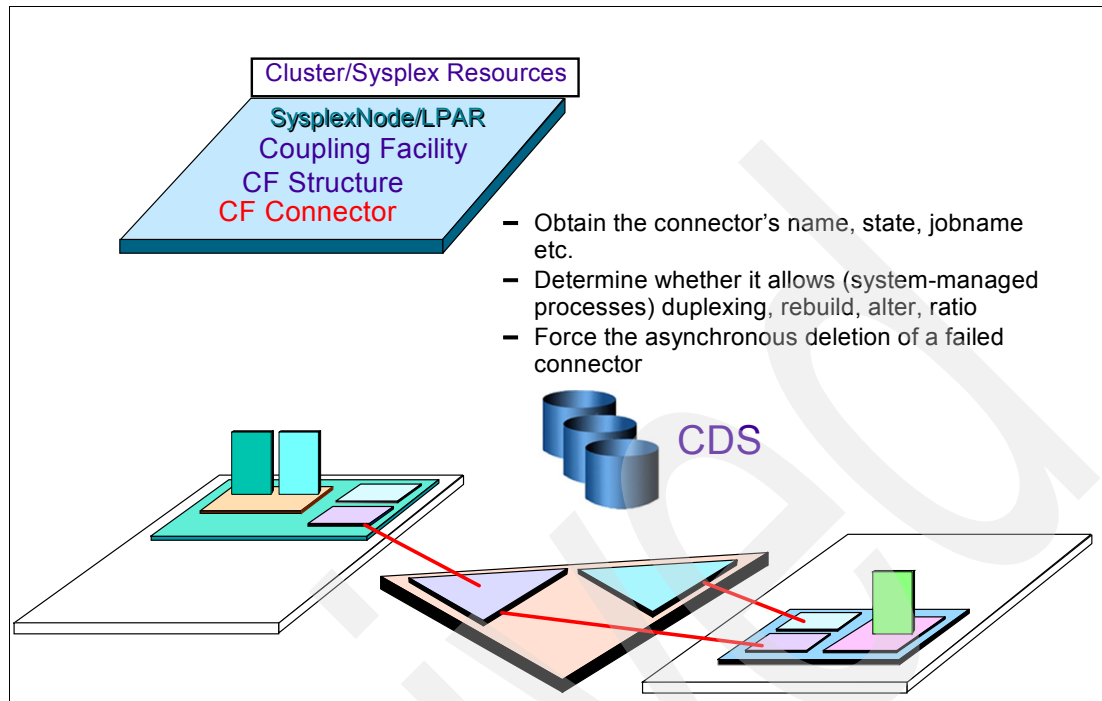


Figure 19-17 CIM instrumentation: CF connector

The cluster CDS resources class provides instrumentation for coupling function, as shown in Figure 19-18.

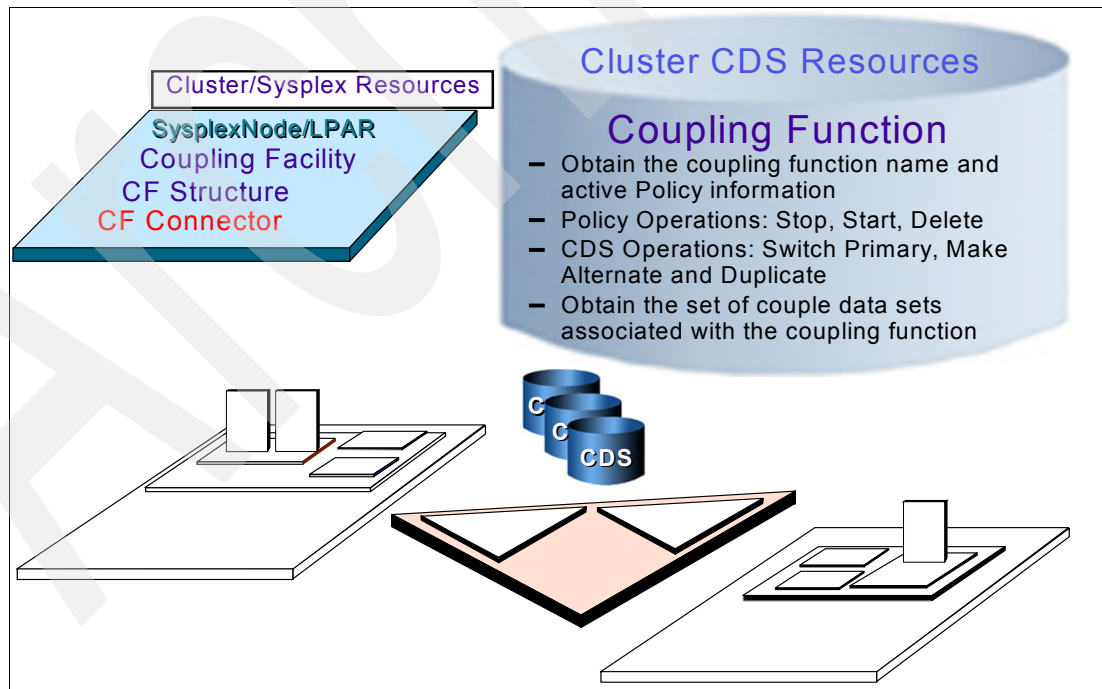


Figure 19-18 CIM instrumentation: coupling function

Other subclasses of Cluster CDS Resources are instrumented as depicted in Figure 19-19 on page 335.

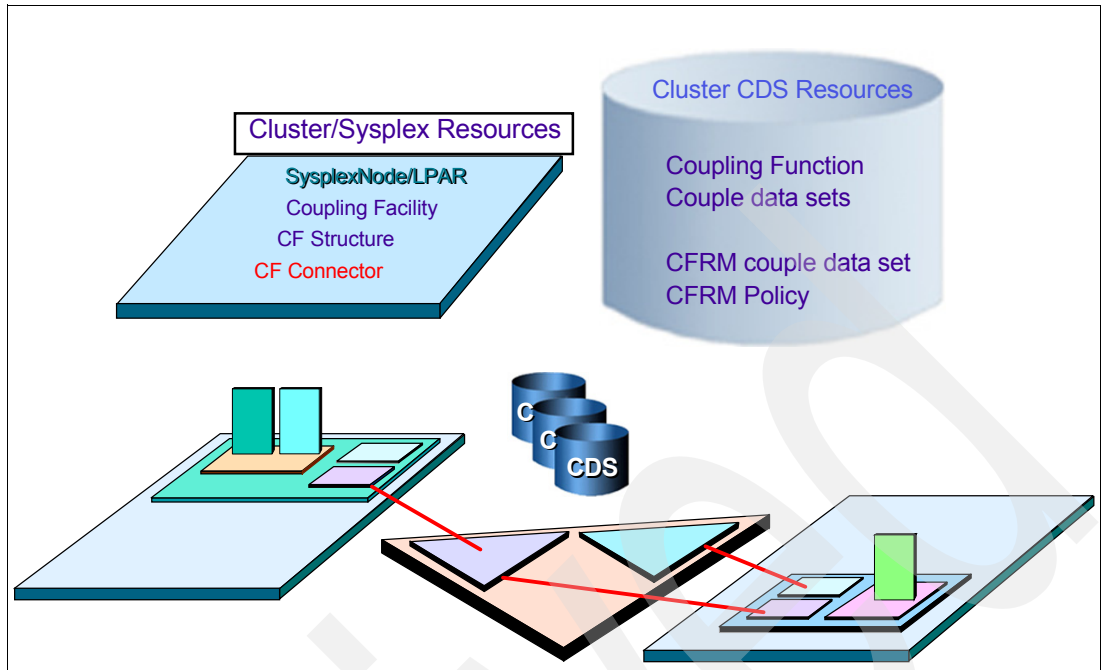


Figure 19-19 CIM instrumentation: sysplex subclasses

### 19.8.5 DFSMSrmm CIM provider

The following DFSMSrmm CIM classes are added with this release:

- ▶ IBMrmm\_LogicalMedia (abstract class)
- ▶ IBMrmm\_PhysicalMedia (abstract class)
- ▶ IBMrmm\_StorageMediaLocation (abstract class)
- ▶ IBMrmm\_Identity (abstract class)
- ▶ IBMrmm\_Product (main class)
- ▶ IBMrmm\_PolicyRule (main class)
- ▶ BMrmm\_Control (main class)

The main classes now offer the `invokeMethod` function to request a list of DFSMSrmm resources by name. This function is particularly useful when a client wants to use the CONTINUE feature to obtain chunks of data, rather than to receive a huge collection of objects at once. It is strongly recommended that you use this function, in conjunction with the CONTINUE operand, to obtain a large number of objects.

All aspects of a volume to a location or shelf location are now modelled by various new association classes. Volume chains and PolicyRule (VRS) chains are also supported with release V1R9.

### 19.8.6 RMF CIM monitoring

Cross-platform management intends to have the same performance metrics be used on various IBM server platforms, including z/OS, Linux (on z), i5/OS, and so on, whenever possible.

Meanwhile, platform-specific extensions such as zAAP or zIIP support, whenever required, are still using the standard-based API and network transport mechanisms.

New metrics have been added to RMF CIM in V1R9 in order to analyze more detailed CPU consumption (including speciality engines) and disk drive or volume statistics.

In order to express WLM-related details, DMTF advanced metrics Breakdown dimensions are exploited.

RMF CIM providers have a prerequisite on the following address spaces:

- ▶ RMF (on every system)
- ▶ RMFGAT (on every system)
- ▶ GPMSERVE (once per sysplex)
- ▶ CFZCIM (on every system) - where RMF CIM is located

As in previous releases, it is recommended that you integrate those address spaces in the automation policy so they are always available.

**Note:** The source code of a client monitoring application is delivered with RMF in SYS1.SAMPLIB(ERBWBEM1. You can compile and run this program with the appropriate Pegasus library. Detailed information about its function and usage is contained within the code.

### New resource class IBMz\_CEC

This resource represents the System z system including processors, memory, I/O drawers, machine, power supplies, and so on; see Figure 19-20. It is a subclass of CIM\_ComputerSystem, which is returned only if RMF is installed.

<i>Property</i>	<i>Type</i>	<i>Description</i>
Name (key)	String	CEC/CPC serial number
CreationClassName (key)	String	Value "IBMz_CEC"
Machine Family	String	Like "2094"
Machine Type	String	Like "715"
Capacity	uint32	System capacity in MSU/hour
ConfiguredPartitions	uint16	Number of configured LPARs of that CEC

Figure 19-20 IBMz\_CEC

Following are the metrics associated to the IBMz\_CEC class:

- ▶ TotalCPTimePercentage – CEC level overall CP utilization time percentage (physical)
- ▶ TotalIIPTimePercentage, TotalAAPTTimePercentage: same for zIIPs and zAAPs
- ▶ TotalSharedCPTimePercentage, TotalSharedIIPTimePercentage, TotalSharedAAPTTimePercentage, TotalSharedICFTimePercentage, TotalSharedIFLTimePercentage: CEC level overall utilization percentages for shared engines only
- ▶ NumberOfSharedCPs, NumberOfSharedAAPs, NumberOfSharedIIPs
- ▶ NumberOfDefinedCPs, NumberOfDefinedAAPs, NumberOfDefinedIIPs, NumberOfDefinedICFs, NumberOfDefinedIFLs



- ▶ NumberOfDedicatedCPs, NumberOfDedicatedAAPs, NumberOfDedicatedIIPs
- ▶ SumOfCPsAcrossLPARs: number of shared logical CPs. For example, if you have four shared CPs in a CEC and they are shared between 10 partitions, with all four being defined to all 10 partitions, then SumOfCPsAcrossLPARs is 40.
- ▶ SumOfIIPsAcrossLPARs, SumOfAAPsAcrossLPARs, SumOfOnlineCPsAcrossLPARs, SumOfOnlineAAPsAcrossLPARs, SumOfOnlineAAPsAcrossLPARs
- ▶ LPARWeightForCP, LPARWeightForIIP, LPARWeightForAAP, LPARWeightForICF, LPARWeightForIFL: total of defined weights on CEC level

### New resource class IBMz\_ComputerSystem

This resource represents a System z LPAR; see Figure 19-21. Like IBMz\_CEC (representing LPARs) and IBMzOS\_ComputerSystem (representing the home operating system), it is a subclass of CIM\_ComputerSystem.

<i>Property</i>	<i>Type</i>	<i>Description</i>
Name	string	LPAR name
CECName	String	Name of the CEC this LPAR exists on
InitialCappingForCP	uint16	enabled / disabled / unknown
InitialCappingForAAP	uint16	enabled / disabled / unknown
InitialCappingForIIP	uint16	enabled / disabled / unknown
WeightManagement	uint16	enabled / disabled / unknown
VaryCPUManagement	uint16	Enabled / disabled / unknown

Figure 19-21 IBMz\_ComputerSystem

Following are the metrics associated to IBMz\_ComputerSystem class:

- ▶ PartitionDefinedCapacityUsedPercentage – percentage of defined MSUs actually used by the given LPAR
- ▶ PartitionDefinedCapacity - MSU capacity of the LPAR
- ▶ PartitionCapacityFourHourAverage – MSU 4-hour rolling average
- ▶ PartitionCapacityCappedPercentage – percentage of time WLM soft-capped the partition
- ▶ TotalCPTimePercentage, TotalIIPTimePercentage, TotalAAPTimePercentage – physical CPU utilization percentages for engines type CP, zIIP, zAAP
- ▶ NumberOfDefinedCPs, NumberOfDefinedIIPs, NumberOfDefinedAAPs, NumberOfDefinedICFs, NumberOfDefinedIFLs, NumberOfDefinedIIPs
- ▶ NumberOfOnlineCPs, NumberOfOnlineIIPs, NumberOfOnlineAAPs, NumberOfOnlineICFs, NumberOfOnlineIFLs
- ▶ NumberOfSharedCPs, NumberOfSharedIIPs, NumberOfSharedAAPs
- ▶ NumberOfDedicatedCPs, NumberOfDedicatedIIPs, NumberOfDedicatedAAPs
- ▶ TotalAAPonCPTimePercentage, TotalIIPonCPTimePercentage
- ▶ LPARWeightForCP, LPARWeightForIIP, LPARWeightForAAP, LPARWeightForIFL

## Additional metrics associated to IBMzOS\_OperatingSystem resources

Additional metrics have been added to show specific resources of the z/OS world (without Breakdown Dimensions):

- ▶ InternalViewIIPTimePercentage, InternalViewAAPTTimePercentage: MVS view of utilization percentage of zAAPs and zIIPs
- ▶ TotalIIPTimePercentage, TotalAAPTTimePercentage: total utilization percentage of zIIPs/zAAPs defined for this OperatingSystem image

### Breakdown dimensions

Some metrics have a breakdown dimension of WLM Service Class Period in the IBMzOS\_BaseMetricDefinition.BreakdownDimensions property. If you retrieve metric values for those definitions, you get back multiple instances with different breakdown values, with IBMzOS\_BaseMetricValue.BreakdownDimension being set to the WLM Service Class Period.

For example, Local Performance Index with the WLM Service Class Period breakdown dimension may retrieve metrics as follows:

- ▶ BreakdownValue = "GOLDSCP.1", MetricValue = "0.7"
- ▶ BreakdownValue = "DONTCARE.4", MetricValue = "5.0"
- ▶ BreakdownValue = "WAS.1", MetricValue = "1.2"

## 19.9 Migration and coexistence considerations

If you are migrating from z/OS V1R8:

- ▶ An automatic repository upgrade to DMTF CIM Schema Version 2.11 will be done at CIM Server startup, issuing the following messages:

```
CFZ12552I: Starting automatic repository upgrade.
```

```
CFZ12563I: Automatic repository upgrade completed successfully.
```

If you are migrating from z/OS V1R7:

- ▶ Use the CFZRCUST migration job on the IPLed target system.

If you are migrating from a z/OS V1R8 CIM server to a z/OS V1R9 CIM server:

- ▶ The migration of the CIM repository is done automatically by the CIM server at its first startup. However, you must perform the following important action:
  - In the environment file /etc/wbem/cimserver\_planned.conf, ensure that the logdir, enableNamespaceAuthorization, and httpAuthType properties do *not* exist. If they exist, delete them.

The contents of z/OS RMF CIM V1R9 are rolled back to z/OS V1R7, with an SPE for z/OS CIM.

### 19.9.1 General migration considerations

Generally, the following steps should be done in order to take benefit of the new functionalities:

- ▶ Update CFZCIM started task procedure

- ▶ Update /etc/wbem/cimserver.env
  - Copy /usr/lpp/wbem/cimserver.env
- ▶ Update /etc/wbem/cimserver\_planned.conf
  - Remove logdir, enableNamespaceAuthorization and httpAuthType properties or...
  - Copy /usr/lpp/wbem/cimserver\_planned.conf
- ▶ Add ARM support
- ▶ Configure SSL certificate based authentication
- ▶ Configure AT-TLS
- ▶ Activate syslogd (if not already active)

## 19.9.2 Cloning considerations

In a context with a sysplex-wise single hierarchical structure deployed, proper settings should be done regarding some of the file systems:

- ▶ A file system (the one mounted on /usr/lpp/wbem) is read-only and sysplex-wide unique
- ▶ The other file systems are read-write and a specific instance is mandatory for each system:
  - For each target system: create its own /etc/wbem directory and /var/wbem
  - Place the /var/wbem on an extra file system data set (about 150MB zFS)
- ▶ Special attention should be given in preserving the file tags (CCSID=ISO8859-1) of the repository.

Figure 19-22 illustrates cloning considerations.

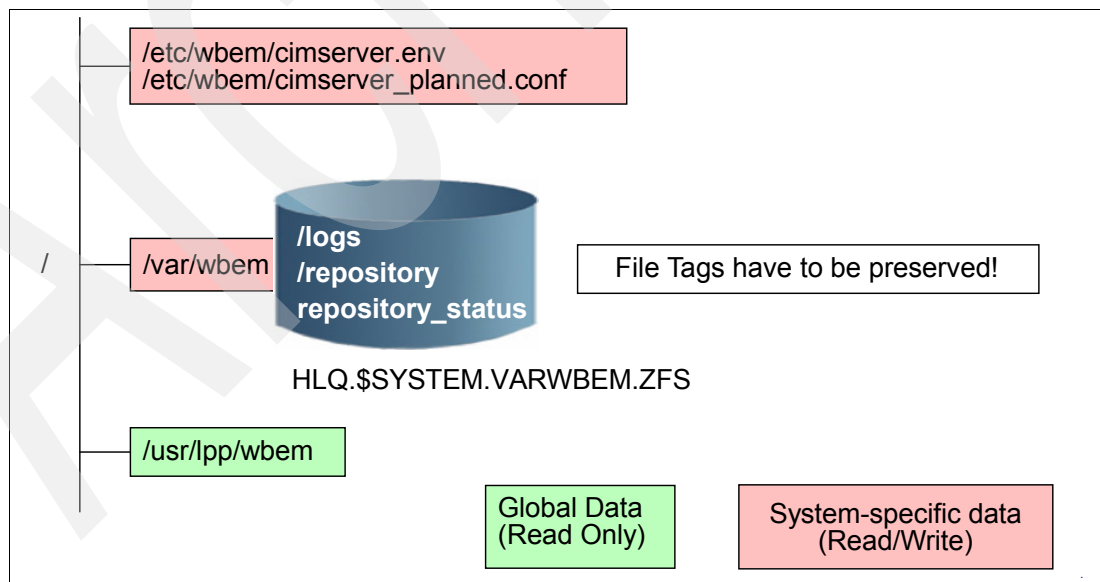


Figure 19-22 Cloning considerations

Archived



## Program management enhancements

This chapter discusses the enhancements introduced in z/OS V1R9 to the program management Binder.

In this chapter we describe the following:

- ▶ Two new Binder control options: MODMAP and INFO
- ▶ Changes to the CHANGE and REPLACE control statements
- ▶ SYSLMOD DD record format verification
- ▶ Binder C/C++ API
- ▶ Support for side-deck files in archive libraries
- ▶ Fast data access services enhancements

## 20.1 New Binder options

The z/OS V1R9 program management Binder is enhanced with two new Binder options: the MODMAP Binder option and the INFO Binder option.

### 20.1.1 The MODMAP Binder option

Obtaining needed information about entry points and other external symbols in a program object often requires many calls to the Binder API. Applications can avoid this for load modules by reading the PDS member directly and find most of the needed information in the CESD. However, this option is not available for programs stored in a PDSE or in UNIX files.

The z/OS V1R9 Binder provides a module map. The Binder module map is used to store the information in the module in a manner that can be read directly or retrieved with a single Binder API call. When it exists, the module map is part of the saved module.

The creation of the module map is controlled by the MODMAP Binder option. The syntax of the MODMAP option is shown in Figure 20-1.

```
MODMAP(NO|LOAD|NOLOAD)
```

Figure 20-1 Syntax of the MODMAP Binder option

<b>NO</b>	Indicates no module map is to be created by the Binder. This is the default value.
<b>LOAD</b>	Indicates the Binder should build the module map in a loadable class. This is supported for both program objects (all formats) and load modules.
<b>NOLOAD</b>	Indicates the Binder should build the module map in a non-loadable class. This is supported only for program objects.

**Note:** When a strong reference to IEWBMP exists in the module, it is regarded as the equivalent to specifying MODMAP(Load), unless MODMAP(NO) is specified.

When the module map is in a loadable class, it is accessed in the following ways:

- ▶ By the IEWBMP address constant.
- ▶ The Binder builds entry IEWBMP for program objects that contain multiple text classes or user-defined classes. If IEWBMP exists, the entry for class B\_MODMAP, which is the module map class, will contain the address of that class in the loaded module.
- ▶ The module map is located at the end of the first segment, that is, the segment containing the entry points. The last double word of the module map contains the offset of the beginning of the module map from the load point of the segment.

When the module map is in a non-loadable class, it is accessed using the Binder API GETD call. The section name is IEWBMP and the class name is B\_MODMAP.

Module map entries are organized in a hierarchy of segment, class, section and entry or segment, class, section and part. For each class, a class entry is followed by section entries for all sections which contribute to the class. Section entries correspond to ED ESD entries in the module. Each section entry is followed by entry points entries (corresponding to LD ESD records) or parts entries (corresponding to PD ESD records) in that section and class. The entries are ordered by segment and then offset within segment.

## Coexistence

APAR OA17827 handles the case when a module with a module map is rebound on a pre-z/OS V1R9 system. In that case, the APAR causes the pre-z/OS V1R9 system to delete the module map from the module.

### 20.1.2 The INFO Binder option

The INFO Binder option is a new option in z/OS V1R9 that provides a way to report service applied to the Binder, such as PTFs and USERMODs, as part of the Binder listing. Use the INFO Binder option when it is required to determine the level of the Binder being executed in a given job beyond the Binder release level. The syntax of the INFO option is shown in Figure 20-2.

```
INFO=NO|YES
```

Figure 20-2 Syntax of the INFO Binder option

- NO** Specifies that a level information report is not to be included in the Binder listing. This can also be specified as NOINFO. This is the default option.
- YES** Specifies that a level information report will be included in the Binder listing.

The report is produced at the end of the Binder listing, just before the message summary. An example of the Binder level information report is shown in Figure 20-3.

```
*** START BINDER LEVEL INFORMATION ***
MODULE   COMPILE DATE   PTF LEVEL
IEWBACTL 06293          UA10162
IEWBBARN 06293          UA15580
IEWBBBIE 06293          UA20277
IEWBBCDS 06293          UA20277
*** END BINDER LEVEL INFORMATION ***
```

Figure 20-3 Example of the Binder level information report

Only sections in the main Binder module IEWBIND are reflected in the level information report.

## 20.2 Enhanced Binder control statements

The CHANGE statement causes an external symbol to be replaced by the symbol in parentheses following the external symbol. The external symbol to be changed can be a control section name, a common area name, an entry name, an external reference, or a pseudo register. More than one such substitution can be specified in one CHANGE statement.

The REPLACE statement is used to replace or delete external symbols. The external symbol can name a section, an entry point, an external reference, or a pseudo register. One section can be replaced with another. All references within the input module to the old section are changed to the new section. Any external references to the old section from other modules are unresolved unless changed.

Prior to z/OS V1R9, both the CHANGE and REPLACE Binder control statements had to precede the module containing the external reference replaced, or the INCLUDE Binder control statement specifying that module. This was due to the fact that the scope of the CHANGE and REPLACE statements was across the next object module, load module, or program object.

The CHANGE and REPLACE statements behavior prior to z/OS V1R9 made it difficult to handle multi-object input files. Many times it was required to split apart or edit the multi-object input.

In z/OS V1R9, a new **-IMMED** optional parameter is added to both the CHANGE and REPLACE Binder control statements. The new parameter affects CHANGE and REPLACE in the same manner. When specified, it indicates that the scope of the CHANGE or REPLACE statements is across the current object module, load module or program object. Therefore, they should not be placed before the module or INCLUDE statement specifying the module, but rather *after* them.

The new syntax of the CHANGE statement is shown in Figure 20-4.

```
CHANGE [-IMMED,]externalsymbol (newsymbol) [,externalsymbol (newsymbol)]...
```

Figure 20-4 Syntax of the CHANGE Binder control statement

The new syntax of the REPLACE statement is shown in Figure 20-5.

```
REPLACE [-IMMED,]externalsymbol1[(externalsymbol2)]...
```

Figure 20-5 Syntax of the REPLACE Binder control statement

**Note:** The **-IMMED** option is not allowed during autocall processing.

### Example

Figure 20-6 shows an example of using the REPLACE Binder control statement with the **-IMMED** option to delete section WTO1DEL from module WTO1.

```
//SYSLIB DD DISP=SHR,DSN=PELEG.LOAD
//SYSLIN DD *
INCLUDE SYSLIB(WT01)
REPLACE -IMMED,WTO1DEL
ENTRY WTO1
NAME WTO1(R)
/*
```

Figure 20-6 Example of using the REPLACE -IMMED Binder control statement

## 20.3 SYSLMOD record format verification

Starting with z/OS V1R9, the Binder does not allow saving a module into a SYSLMOD data set of type PDS unless the data set's record format is undefined (RECFM=U). This enhancement should help to prevent accidentally saving modules into the wrong PDS. When the Binder saves a module to a PDS that did not have RECFM=U, the PDS's RECFM is



changed to RECFM=U which may cause the other members in the PDS to become unreadable.

With z/OS V1R9, when the Binder tries to save a module into a PDS with record format other than RECFM=U, it ends with return code 12 and does not save the module. In addition, the following message is issued:

```
IEW2735S DAOF OUTPUT DATA SET FOR DDNAME SYSLMOD HAS INVALID RECORD FORMAT.  
RECFM=U IS REQUIRED.
```

For PDSE data sets, this is already the case prior to z/OS V1R9.

If you still want to deliberately save a module into a PDS data set with record format other than RECFM=U, you must specify RECFM=U on the SYSLMOD DD card in the JCL. When RECFM=U is specified in the JCL, the Binder honors the request and saves the module, even if the original PDS was not allocated with RECFM=U.

## 20.4 Binder C/C++ API

The Binder provides an API to assembler programs in the form of the IEWBUFF and IEWBIND macros. The IEWBUFF macro is used to generate, initialize, and map the data buffers required during binder processing. The IEWBIND macro is used to call the Binder functions. Prior to z/OS V1R9, it was possible to call the Binder API from high level languages such as C and C++, but the caller had to provide a certain environment, buffers, and parameter lists. This made it very difficult to use the Binder API in high level languages.

Starting with z/OS V1R9, the Binder provides an API to C/C++ programs. The Binder's C/C++ API is a C code interface that implements the initialization and access functions of the Binder API and fast data access services. The Binder C/C++ API saves you the need to manage buffers in your application. The Binder C/C++ API manages the buffer storage automatically within the API. The Binder C/C++ API also loads and deletes Binder interface modules automatically.

The Binder C/C++ API provides a header file named `__iew_api.h` to allow C/C++ programs to interface with the Binder API and fast data access services in a standard way. The header file is installed in the `/usr/include/z/OS UNIX` directory. The header file includes:

- ▶ API data types

The API data types are used by API access and utilities functions. They allow for data encapsulation and a convenient way for parameter passing.

- ▶ Buffer header and entry definition for each buffer type

The buffer header and buffer entry definition for each buffer type are based on the version 6 of API buffer formats as described in Appendix D of *z/OS MVS Program Management: Advanced Facilities*, SA22-7644. An application can use the appropriate buffer type to cast the return buffer after calling some API access functions.

- ▶ Binder API and fast data access services access functions

Access functions allow users to access the Binder API functions.

- ▶ Binder API and fast data access services utilities functions

Utilities functions allow users to access return values and enable the creation of Binder API buffers for API calls that require buffers as input.

The actual code for the Binder C/C++ API is provided in a non-XPLink DLL installed in the /usr/lib z/OS UNIX directory. Under /usr/lib you can find the side deck file iewbndd.x and the DLL file iewbndd.so.

**Note:** The Binder C/C++ API header file, side deck file, and DLL file are only installed in the z/OS UNIX environment.

Table 20-1 lists the Binder API functions provided by the Binder C/C++ API.

Table 20-1 Binder C/C++ API functions

Binder API	Binder C/C++ API function
Add alias	__iew_addA()
Align text	__iew_alignT()
Alter workmod	__iew_alterW()
Perform incremental autocall	__iew_autoC()
Bind workmod	__iew_bindW()
Close workmod	__iew_closeW()
Get compile unit list	__iew_getC()
Get data	__iew_getD()
Get ESD data	__iew_getE()
Get names	__iew_getN()
Import a function or external variable	__iew_import()
Include module via DEPTR (BLDL)	__iew_includeDeptr()
Include module via NAME	__iew_includeName()
Include module via POINTER	__iew_includePtr()
Include module via SMDE	__iew_includeSmde()
Include module via DEPTR (CSVQUERY)	__iew_includeToken()
Insert section	__iew_insertS()
Load workmod	__iew_loadW()
Open workmod	__iew_openW()
Order sections	__iew_orderS()
Put data	__iew_putD()
Rename symbolic references	__iew_rename()
Reset workmod	__iew_resetW()
Save workmod	__iew_saveW()
Set library	__iew_setL()
Set option	__iew_setO()
Start segment	__iew_startS()

Table 20-2 lists the fast data access services functions provided by the Binder C/C++ API.

Table 20-2 Binder C/C++ API fast data access functions

Fast data access service	Binder C/C++ API function
End a session	__iew_fd_end()
Get compile unit list	__iew_fd_getC()
Get data	__iew_fd_getD()
Get ESD data	__iew_fd_getE()
Get names	__iew_fd_getN()
Open a session	__iew_fd_open()
Starting a session with BLDL data	__iew_fd_startDcb()
Starting a session with a System DCB	__iew_fd_startDcbS()
Starting a session with a DD name or path	__iew_fd_startName()
Starting a session with a CSVQUERY token	__iew_fd_startToken()

A detailed example of using the Binder C/C++ API is provided in *MVS Program Management: Advanced Facilities*, SA22-7644.

## 20.5 Support for side deck definition files in archive files

The Binder supports autocall from z/OS UNIX archive libraries. These archive libraries may contain members that are object files in OBJ, XOBJ, and GOFF format and with special directory information similar to that contained in C370LIB object libraries. Archive libraries are created by the z/OS UNIX **ar** command. While the **ar** command is typically used to create archive libraries of object files, it can also be used to create archive libraries of non-object files, or archive libraries containing a combination of object files and non-object files.

In z/OS V1R9, the Binder is able to process non-object files which are side files with **IMPORT** control statements, or other files of control statements, that are contained in a z/OS UNIX archive library.

To be processed by the Binder, non-object files must have the following characteristics:

- ▶ For members that are side files (containing **IMPORT** control statements), an **IMPORT** statement must be the very first statement in the file, in the same format produced by the Binder when it writes to the **SYSDEFSD** DD.
- ▶ For members that are files specifically identified as containing Binder control statements, the first statement must contain the string **\*!** in the first two columns, followed by the string **IEWBIND INCLUDE**. These two strings may be separated by blanks, but must be contained in a single statement.

Adding the **\*! IEWBIND INCLUDE** string to a file containing Binder control statements is a way to generalize the Binder's support for side files to files containing other control statements. The Binder supports side files in archive libraries even if they do not begin with the **\*! IEWBIND INCLUDE** string, as long as the **IMPORT** statement is the first statement in the file.

Figure 20-7 on page 348 shows an example of how to add a comment to a side deck definition file contained in an archive library.

```
*! IEWBIND INCLUDE
*
* my side-file for my-product
*
IMPORT CODE,'myd11','mylib2func'
```

Figure 20-7 Example of a side deck file contained in a z/OS UNIX archive library

A non-object file must be positioned as the first member of the archive library, not including the symbol table member `__SYMDEF`. The Binder processes that member as if it had been explicitly included as binder input. In case several non-object files are contained in the archive library, at least one of those files must be positioned as the first member of the archive library. The Binder processes that first member and then includes any other such members that are included in the archive library.

## 20.6 Binder fast data access enhancements

The Binder fast data access services provide a more efficient alternative for a subset of the Binder API functionality. Fast data access services allow you to inspect and obtain information about program object only. Internally, fast data access does not provide recovery (ESTAE or ESPIE) and does not rebuild program objects to a common internal format, thus providing better performance. Fast data access is optimized for minimum overhead when reading program object residing on DASD.

Starting with z/OS V1R9, the Binder's fast data access services are changed to exploit the 64-bit address space for internal data, instead of a data space. To use the fast data access services in z/OS V1R9, the program must run with `MEMLIMIT>0` set in the `SMFPRMxx` parmlib member, `JCL`, or the `IEFUSI` exit. Otherwise, the Binder fails with return code 12, reason code 1080002F and the following message is issued:

```
IEW2091S IEWBFDAT UNABLE TO GET nnnn MB OF 64-BIT STORAGE, IARV64 CODE xxxx
```

In addition, in z/OS V1R9, fast data access services provide a trace facility. The trace is controlled by the existence of the `IEWTRACE DD`. No particular DCB parameters are required for the `IEWTRACE DD` and it can be allocated to a `SYSOUT` or a sequential data set. The trace is intended for IBM use.

Figure 20-8 on page 349 shows an example of the trace output.

```

Trace format: (mod,proc,id,v1) (w2) (w3)
proc: see TRACE ENTRIES in module listing
id=00: ENT, v1 is nr. of parms in w2 and/or w3
id=FF: XIT, v1=retcode, w2=reason, w3=returnaddr
else if v1=08: CHR, w2,w3 is text
else: INT, v1 is nr. of words traced in w2 and/or w3

----- IEWBQINI = 205050C8
02010000 00000000 00000000 qini entered
02010101 00000000 00000000 getmain return code
----- IEWBQRED = 20501E10
03010001 00000001 00000000 qred entered, parm=1
03020002 C2C2D6D6 D9C24040 open_sb for BB00RB
03100000 00000000 00000000 create_smde entered
03100102 00000000 00000000 deserv rc=0, rs=0
0310FF00 00000000 20502524 return 0 to 20502524
0302FF00 00000000 20501F78 return 0 to 20501F78
03130000 00000000 00000000 get_po_len entered
03090000 00000000 00000000 read_bpam entered
03090101 00000000 00000000 read offset = 0
03090208 C9C5E6D7 D3D4C840 P0+0 contains 'IEWPLMH '
0309FF00 00000000 20504A18 return 0 to 20504A18
0313FF00 00000000 20501FA6 return 0 to 20501FA6
0301FF00 00000000 20505346 return 0 to 20505346

```

Figure 20-8 Fast data access trace example

Archived

## JES2 and JES3 enhancements

This chapter discusses the changes introduced to JES2 and JES3 in z/OS V1R9.

For JES2, the following topics are discussed:

- ▶ SSI requests authorization enhancements
- ▶ \$C Job command enhancements
- ▶ \$TRACE facility enhancements
- ▶ Changes to JES2 installation exits

For JES3, the following topics are discussed:

- ▶ Relief of the OSE buffer number limit
- ▶ More efficient use of spool space

## 21.1 JES2 enhancements

z/OS V1R9 has enhancements to JES2 in the following areas.

- ▶ SSI updates to these SSI codes:
  - Destination Validation/Conversion (11)
  - SWB Modify (70)
  - JES2 router SSI (71) to do the following:
    - Obtain/Return JOBCLASS information
    - Spool read
    - Convert device id (DEVID) to device name
    - Obtain/Return monitor information
  - Notify User (75)
  - SAPI (79)
  - Unauthorized spool browse (81)
- ▶ Cancel jobs during job select

A cancel job (**\$CJ**) has been extended to be possible during job select.
- ▶ \$TRACE enhancements for better diagnostic data
- ▶ \$DSPL performance

Logic for the \$DSPL command was extremely inefficient when displaying status of DRAINING volumes, possibly causing noticeable performance degradation.
- ▶ Stunted volume processing

When starting a spool volume, there may not be enough room in the track group map for the entire volume. Operators are given a chance to decide if they want to use as much as possible space for the volume or not via the HASP811 (**\$SSPL** command) or HASP853 (initialization) message. If they reply YES, then that is what we call a *stunted volume*. It will remain that way unless the volume is drained and restarted.

Beginning With z/OS V1R9, JES2 detects stunted volumes when appropriate and expands them to full defined capacity.

## 21.2 SSI requests authorization enhancements

JES2 supports a rich set of APIs that allows applications to manage jobs and SYSOUT data sets. This API is implemented through the subsystem interface (SSI). Prior to z/OS V1R9, most of the JES2 SSI requests required callers to be authorized, running in supervisor state, or by system key. This requirement poses a security threat to the system because authorized callers can bypass system security and potentially harm the system. It was up to callers to ensure that they did not compromise the system.

JES2 in z/OS V1R9 addresses this security issue by enhancing much of the authorization requirements for the SSI requests it supports. Many JES2 SSI requests are updated in z/OS V1R9 to allow unauthorized callers. JES2 SSI requests are changed to access data passed by the caller in the caller's key. This is now done for unauthorized callers as well as authorized callers. Moreover, RACF authorization checks are added to protect sensitive JES2 data that is returned by some of the JES2 SSI requests.



Unlike authorized callers who run with system key, mistakes or errors in an unauthorized caller application might cause S0C4 ABENDs because they are not be able to access some storage areas or pass bad storage areas to JES2 SSI. Such an S0C4 ABEND causes JES2 to issue an SVC dump even though the S0C4 ABEND is actually an application error and is not interesting to JES2. Therefore, JES2 recovery routines are designed to internally convert S0C4 ABENDs that occur when accessing user passed data to a new ABEND code S1E0. S1E0 ABENDs should be regarded as application error and not a system or JES2 error.

For more information about SSI see *MVS Using the Subsystem Interface*, SA22-7642.

### 21.2.1 SSI 11 - User destination validation/conversion service

The user destination validation/conversion service (SSI function code 11) provides a requesting program with the ability to convert or validate a remote destination.

SSI 11 does not functionally change in z/OS V1R9. The only change is that SSI 11 now supports unauthorized callers. The storage passed by the caller (such as the SSOB, SSUS, and so on) is now referenced in the SSI request caller's key.

### 21.2.2 SSI 70 - Scheduler facilities function

The scheduler facilities function (SSI function code 70) provides a requesting program with the ability to modify those characteristics of SYSOUT data sets that are controlled by Scheduler Work Block Text Unit (SWBTU) data. JES2 supports one function type of this SSI function code, SWB Modify. SWB Modify allows a user to modify SWBTU data associated with a SYSOUT data set. SWBTU data consists of dynamic output values such as ADDRESS, CLASS, COPIES, FORMS, NOTIFY, and so on.

The SWB modify service is now available to unauthorized callers. In addition, a new minimal authorization check is implemented for authorized callers only. To request the new authorization check, set bit SSSFSECL in SSSFFLG1. The new option tells JES2 to bypass security checking unless Multi Level Security is active on the system (MLACTIVE set in RACF).

If MLACTIVE is set, then JES2 ensures proper SECLABEL dominance checking instead of the regular JESSPOOL or ISFAUTH check. If an unauthorized caller attempts to request SECLABEL dominance checking, a return code indicating an error is returned to the SSI caller, as follows:

- ▶ SSSFREAS=SSSFNRI, indicating invalid input to request
- ▶ SSSFMREA=SSSFMSCI, indicating invalid security request

Furthermore, extra validation of the SSOB is now being done by SSI 70 and a new reason code is returned as SSSFMREA=SSSFMIVX if the SSOB extension is too small to perform the service.

### 21.2.3 SSI 71 - JES job information services

The JES job information services (SSI function code 71) allow a user-supplied program to obtain information about jobs in the JES2 queues. SSI 71 supports these subfunctions:

- ▶ Spool read service

The service now issues a new RACF call to verify the caller is authorized to read the data on the spool. The new RACF call is always issued for unauthorized callers.

By default, the new RACF call is not issued for authorized callers. However, authorized callers can set the SPIORACF bit in SPIOOPT to force the RACF call. The entity name and class passed on to the RACF call depends on whether the service is able to locate the security token (RTOKEN) associated with the user that owns the control block, as explained here:

- ▶ If the associated security token is located, then a JESSPOOL check will be made passing a profile in the format **node.userid.jobname.jobid.SP00LI0.cbname** and the RTOKEN found.
- ▶ If the associated security token is not found, then a JESJOBS class check is issued passing a profile in the format **SP00LI0.node.jobname.jobid.cbname** and the JES2 address space security token as RTOKEN.

Additional data is added to the HDB, IOT, and SWBIT data areas to help the spool read service locate the security token for these data areas. The new fields are not populated by older releases of JES2, so reading these data areas that were created on a pre-z/OS V1R9 release will default to the JESJOBS class check.

**Note:** The new RACF checks and additional I/O adds a significant overhead to the spool read service. Using the extended status function call (SSI 80), when possible, is more efficient.

- ▶ JES2 health monitor information

In addition to changing this subfunction to support unauthorized callers, the JES2 monitor SSI is updated to return information about JES2 storage usage. The sub function now returns the same information that is returned by the **\$JD DETAILS(STORAGE)** and **\$JD HISTORY(STORAGE)** commands.

Figure 21-1 shows a sample output of the **\$JD DETAILS(STORAGE)** command to demonstrate the information returned by this SSI subfunction.

\$JD DETAILS(STORAGE)						
\$HASP9103 D DETAIL 585						
\$HASP9108 JES2 STORAGE USAGE (PAGES) SINCE 2007.131 15:00:00						
AREA	REGION	USAGE	LOW	HIGH	AVERAGE	
-----						
<16M USER	2,042	249	249	249	249	
<16M SYSTEM	2,042	66	66	67	66	
>16M USER	377,856	65,234	65,234	65,234	65,234	
>16M SYSTEM	377,856	2,601	2,601	2,601	2,601	

Figure 21-1 Output of the **\$JD DETAILS(STORAGE)** command

- ▶ Job class information

The job class information service provides the attributes of a job class. This subfunction does not functionally change in z/OS V1R9. It is only changed to support unauthorized callers.

- ▶ Convert device ID service

The Convert device ID service translates a binary device ID into its EBCDIC device name. This subfunction does not functionally change in z/OS V1R9. It is only changed to support unauthorized callers.

- ▶ Checkpoint version information service

This subfunction is not changed in z/OS V1R9. You must be authorized to call this service.

## 21.2.4 SSI 75 - Notify user message service call

The notify user message service call (SSI function code 75) provides a requesting program the ability to send a message to other users who are either on the same networking node or on another node.

SSI 75 now supports unauthorized callers. The service allows the caller to provide it with a security token to associate with the caller (field SSNUTKNA). It is undesirable to allow unauthorized callers to alter the security environment. Therefore, support for passing of a security token is restricted only to authorized callers. If an unauthorized caller passes a security token to the service, the notify message is sent, but the security token is ignored and the service returns with SSOBRETN=4 and SSNUERCD=40.

In addition, JES2 now supports passing a member name for the SEND command in the SSNUMEMB field. If an invalid member name is specified, SSNUERCD is set to SSNUMEME.

## 21.2.5 SSI 79 - SYSOUT application programming interface (SAPI)

SAPI (SSI function code 79) allows JES to function as a server for applications needing to process SYSOUT data sets residing on JES spool. Use of the SAPI SSI call allows a user-supplied program to access JES SYSOUT data sets independently from the normal JES-provided functions (such as print or network). Users of this function are typically application programs operating in address spaces external to JES. SAPI supports multiple, concurrent requests from the applications' address spaces.

SSI 79 does not functionally change in z/OS V1R9. The only change is that SSI 79 now supports unauthorized callers. The storage passed by the caller (such as the SSOB, SSS2, and so on) is now referenced in the SSI request caller's key.

## 21.2.6 SSI 80 - Extended status function call

The extended status function call (SSI function code 80) allows a user-supplied program to obtain detailed status information about jobs and SYSOUT in the JES queue. Both JES2 and JES3 subsystems support job status information.

SSI 80 now supports unauthorized callers. SSI 80 is also enhanced with a RACF SECLABEL dominance check. The SECLABEL check is optional for authorized callers and mandatory for unauthorized callers. The SECLABEL dominance check is only performed if MACTIVE is set in RACF.

Additionally, a number of new indicator fields are now returned from the service, including:

- ▶ Job is being processed for End of Memory
- ▶ Job is on the JES2 rebuild queue
- ▶ Position of the job on the class queue or phase queue
- ▶ Submitter's security group name
- ▶ APPC TP job name, TP job ID, entry start time and time, execution start time and accounting information

New filters are now supported, including:

- ▶ Select IP or non IP routed SYSOUT
- ▶ Select local vs. network routed SYSOUT

- ▶ Select SYSOUT based on the OUTDISP values (one bit for each of write, hold, keep, and leave)

Furthermore, for performance reasons, the ability to pass multiple values for certain filters is added in z/OS V1R9. The multiple value fields function as OR filters. The multiple value fields apply to job name, job class, job destination and job phase, as well as SYSOUT class and SYSOUT destination.

### 21.2.7 SVC 99 - spool browse

SVC 99 is the dynamic allocation service. SVC 99 also supports non-JCL dynamic allocation functions. The non-JCL dynamic allocation functions are used to ask the system for data set information that has no equivalent JCL parameters. One of the non-JCL dynamic allocation functions represents allocation of a subsystem data set (the DALSSREQ key). JES2 provides support for browsing spool data sets through the SVC 99 non-JCL dynamic allocation function. To use the DALSSREQ key, the caller must be authorized.

Externally, spool browse is not an SSI. However, when an application uses SVC 99 to browse a spool data set, the system issues an SSI 6 request internally to ask JES2 for the service.

In z/OS V1R9, the spool browse service is changed to support unauthorized callers. The DALSSREQ key still requires the caller to be authorized. An unauthorized caller can use a new key, DALUASSR, to request spool browse. When the user uses the DALUASSR key to ask for spool browse, the system issues an SSI 81 request internally, instead of an SSI 6 request. SSI 81's support for spool browse is the same as SSI 6, as on pre-z/OS V1R9 systems.

**Tip:** To change your application to run unauthorized and to use the DALUASSR key instead of DALSSREQ, simply replace the DALSSREQ key with DALUASSR. Both keys have the same count, length, and data.

## 21.3 \$C Job command enhancements

When an initiator is ready to run a new job, it uses the job select SSI request (SSI 5) to request JES2 to scan the job queue and select a new job to run under this initiator. In the early stages of job select processing, while JES2 and MVS are still building the environment required to run the job under the initiator, the initiator is no longer acting like an initiator, but it does not yet have the job that is it about to run.

Therefore, the MVS **CANCEL** and **FORCE** commands cannot act on jobs or initiators in early stages of job selection. When trying to cancel an initiator in this stage, for example using a **C INIT,A=n** command, the following message is issued:

```
IEE341I INIT                NOT ACTIVE IN SPECIFIED ADDRESS SPACE
```

From the system's point of view, the address space is no longer an initiator and thus cannot be canceled or forced using the **CANCEL** or **FORCE** commands. At this stage, only JES2 knows which job is assigned to the initiator.

Starting with z/OS V1R9, the JES2 **\$C Job** command is enhanced to support canceling jobs in early stages of job select. In z/OS V1R9, the JES2 cancel command uses the CALLRTM system service to terminate the job. Requesting a dump is also supported by the **\$C Jnnn,DUMP** command.

When a job is terminated using the **\$C Job** command, the address space ABEND code is S422 and the reason code is X'00010204'. When a job is terminated using the **\$C Jnnn, DUMP** command, the address space ABEND code is S422 and the reason code is X'00020204'.

Figure 21-2 shows an example of terminating a job in early stages of job select using the JES2 **\$C Job** command.

```
$cj18
JOB00018 $HASP890 JOB(JOBT)
$HASP890 JOB(JOBT) STATUS=(EXECUTING/IBM1),CLASS=A,
$HASP890 PRIORITY=9,SYSAFF=(ANY),HOLD=(NONE),
$HASP890 CANCEL=YES
STC00017 IEF450I INIT INIT - ABEND=S422 U0000 REASON=00010204
          TIME=19.36.10
STC00017 $HASP395 INIT ENDED
JOB00018 $HASP310 JOBT TERMINATED AT END OF MEMORY
```

Figure 21-2 Using the **\$C J** command to terminate an address space

However, there are cases where the **\$C Job** command cannot cancel a job. To handle these situations, a new **FORCE** parameter is introduced in z/OS V1R9 for the **\$C Job** command. When **\$C Jnnn, FORCE** is used, JES2 issues the **CALLRTM TYPE=MEMTERM** system service to terminate the initiator address space. The address space is ended with ABEND code S422 and a reason code of X'00030208'.

## 21.4 \$TRACE facility enhancements

The \$TRACE facility allows JES2 diagnostic information to be collected in order to help debugging problems in the system. On large systems, the amount of collected trace information is huge. Sometimes it is difficult to locate the interesting information in the trace output, because the trace is flooded with information that is irrelevant to the current problem.

In z/OS V1R9, it is now possible to specify filters by address space ID, job name, job number and TCB address for the \$TRACE facility. By using the filters, the amount of data written to the trace data set is significantly reduced to include events that occur only for the desired jobs. In addition, the precision of the time stamps in the trace output is increased.

### 21.4.1 TRACE initialization statement and \$T TRACE command

To control trace filters, new keywords are added to the TRACE initialization statement and the \$T TRACE command. The new keywords are:

<b>ASID=</b>	Specifies the ASID used when filtering this JES2 trace point.
<b>JOBNAME=</b>	Specifies the job name used when filtering this JES2 trace point.
<b>JOB_NUMBER=</b>	Specifies the job number used when filtering this JES2 trace point.
<b>TCB_ADDRESS=</b>	Specifies the TCB address in order to further limit tracing to the specified TCB. This is in addition to any filtering that is applied. This operand is ignored if ASID=, JOBNAME=, and JOB_NUMBER= are not specified.

**Note:** The filters do not support masking or generics.

## Using trace filters

The ASID=, JOBNAME=, and JOB\_NUMBER= filters are treated as OR filters by JES2. Trace data is collected if any one of the filters is matched. The TCB\_ADDRESS= filter is treated as an AND filter. When specified, it must always match the current TCB address in the PSA field PSATOLD to allow trace data to be collected. How the ASID=, JOBNAME= and JOB\_NUMBER= filters are applied depends on the current address space and JES2 task in control.

When the JES2 main task or subtasks running in the JES2 address space are in control, the values specified in ASID=, JOBNAME= and JOB\_NUMBER= always refer to the job that JES2 is processing. The PCEASID field is checked for the ASID= filter and the JQE pointed by PCEJQE is checked for the JOBNAME= and JOB\_NUMBER= filters. The TCB address specified on the TCB\_ADDRESS= filter must match the current TCB address in the JES2 address space to allow trace data to be collected.

Outside the JES2 address space, for example in a job running under an initiator, an FSS or an application using SAPI, the ASID=, JOBNAME=, JOB\_NUMBER= and TCB\_ADDRESS= filters apply to the address space where the trace is being taken. The filters are checked against fields in the ASCB and, when available, the SJB.

After the trace filters are set by the **\$T TRACE** command for a trace ID, you can use the **\$\$ TRACE(n)** or **\$T TRACE(n), START=YES** command to start the trace. Use the **\$P TRACE(n)** or **\$T TRACE(n), START=NO** to stop the trace.

## \$TRACE examples

Figure 21-3 on page 359 shows an example of using the new filter keywords to set up a trace. In the example, the first **\$T TRACE** command sets filters for ASID number X'AB' and a job name of PELEGTSO. ASID X'AB' is actually a TSO/E user who is just about to submit a job by the name PELEGTSO.

Use the **START=YES** parameter to start trace IDs 1, 2, 18 and 19 to see all the JQE request processor (JQRP PCE) calls to \$SAVE and \$RETURN and subsequent phases. Our trace does not include input processing since the INTRDR trace is not active. See Figure 21-4 on page 360 for an example that includes INTRDR tracing.

Use the **\$T TRACEDEF** command to start the \$TRACE facility and set the output class and log size.

Use the **\$T TRACEDEF** command to stop the trace. The SPIN parameter tells JES2 to spin off the \$TRCLOG trace data set.

To turn off trace IDs 1, 2, 18 and 19, use the **\$P TRACE** command.

```

$T TRACE(1-2,18-19),START=YES,ASID=AB,JOBNAME=PELEGTSO
$HASP667 TRACE(1)
$HASP667 TRACE(1) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$HASP667 TRACE(2)
$HASP667 TRACE(2) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$HASP667 TRACE(18)
$HASP667 TRACE(18) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$HASP667 TRACE(19)
$HASP667 TRACE(19) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$T TRACEDEF,ACTIVE=YES,LOG=(START=YES,CLASS=A,SIZE=10000)
$HASP698 TRACEDEF
$HASP698 TRACEDEF ACTIVE=YES, TABLES=3, PAGES=9,
$HASP698 TABWARN=80, TABFREE=3, LOG=(CLASS=A,
$HASP698 START=YES, SIZE=10000),
$HASP698 STATS=(TOTDISC=0, DISCARDS=0, IDS=(1,2,
$HASP698 18,19))
$HASP800 $TRCLOG THE JES2 EVENT TRACE LOG IS NOW ACTIVE
$T TRACEDEF,ACTIVE=NO,SPIN
$HASP698 TRACEDEF
$HASP698 TRACEDEF ACTIVE=NO, TABLES=3, PAGES=9,
$HASP698 TABWARN=80, TABFREE=2, LOG=(CLASS=A,
$HASP698 START=YES, SIZE=10000),
$HASP698 STATS=(TOTDISC=0, DISCARDS=0, IDS=(1,2,
$HASP698 18,19))
$HASP801 $TRCLOG JES2 EVENT TRACE LOG QUEUED TO CLASS A (OUTDISP=WRITE)
$P TRACE(1-2,18-19)
$HASP667 TRACE(1) START=NO,ASID=00AB,JOBNAME=PELEGTSO
$HASP667 TRACE(2) START=NO,ASID=00AB,JOBNAME=PELEGTSO
$HASP667 TRACE(18) START=NO,ASID=00AB,JOBNAME=PELEGTSO
$HASP667 TRACE(19) START=NO,ASID=00AB,JOBNAME=PELEGTSO

```

Figure 21-3 Using filters on the \$T TRACE command

## 21.4.2 INTRDR tracing

New keywords are added to the INTRDR initialization statement and **\$T INTRDR** to control internal reader trace filters. These filters are used only when INTRDR TRACE=NO is specified. The filter are treated as OR filters by JES2. Trace data is collected if any one of the filters is matched. When TRACE=YES is set, the filters do not apply and all internal reader PUTs in the system are traced. Generics are not supported for any of the INTRDR filters.

The new keywords are:

- ASID\_TRACE=** Specifies the ASID used when filtering this internal reader.
- JOBNAME\_TRACE=** Specifies the job name used when filtering this internal reader.
- JOB\_NUMBER\_TRACE=** Specifies the job number used when filtering this internal reader.

## INTRDR trace examples

Figure 21-4 shows an example of using the \$TRACE facility and setting internal reader filters. After the filters for the internal reader are defined, turn on the selective trace IDs 11 and 12 to trace \$SAVE and \$RETURN for the internal reader.

```
$T TRACE(1-2,18-19),START=YES,ASID=AB,JOBNAME=PELEGTSO
$HASP667 TRACE(1)
$HASP667 TRACE(1) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$HASP667 TRACE(2)
$HASP667 TRACE(2) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$HASP667 TRACE(18)
$HASP667 TRACE(18) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$HASP667 TRACE(19)
$HASP667 TRACE(19) START=YES,ASID=00AB,
$HASP667 JOBNAME=PELEGTSO
$T INTRDR,TRACE=NO,ASID=AB,JOBNAME=PELEGTSO
$HASP838 INTRDR
$HASP838 INTRDR AUTH=(DEVICE=YES,JOB=YES,SYSTEM=YES),
$HASP838 BATCH=YES,CLASS=A,HOLD=NO,HONORLIM=NO,
$HASP838 PRTYINC=0,PRTYLIM=15,SYSAFF=(ANY),
$HASP838 TRACE=NO,ASID_TRACE=00AB,
$HASP838 JOBNAME_TRACE=PELEGTSO
$T TRACE(11-12),START=YES
$HASP667 TRACE(11) START=YES
$HASP667 TRACE(12) START=YES
$T TRACEDEF,ACTIVE=YES,LOG=(START=YES,CLASS=A,SIZE=10000)
$HASP698 TRACEDEF
$HASP698 TRACEDEF ACTIVE=YES,TABLES=3,PAGES=9,
$HASP698 TABWARN=80,TABFREE=3,LOG=(CLASS=A,
$HASP698 START=YES,SIZE=10000),
$HASP698 STATS=(TOTDISC=0,DISCARDS=0,IDS=(1,2,
$HASP698 11,12,18,19))
$T TRACEDEF,ACTIVE=NO,SPIN
$HASP698 TRACEDEF
$HASP698 TRACEDEF ACTIVE=NO,TABLES=3,PAGES=9,
$HASP698 TABWARN=80,TABFREE=2,LOG=(CLASS=A,
$HASP698 START=YES,SIZE=10000),
$HASP698 STATS=(TOTDISC=0,DISCARDS=0,IDS=(1,2,
$HASP698 11,12,18,19))
$P TRACE(*)
$HASP667 TRACE(1) START=NO,ASID=00AB,JOBNAME=PELEGTSO
$HASP667 TRACE(2) START=NO,ASID=00AB,JOBNAME=PELEGTSO
$HASP667 TRACE(11) START=NO
$HASP667 TRACE(12) START=NO
$HASP667 TRACE(18) START=NO,ASID=00AB,JOBNAME=PELEGTSO
$HASP667 TRACE(19) START=NO,ASID=00AB,JOBNAME=PELEGTSO
```

Figure 21-4 Using filters on the \$T INTRDR command



## 21.5 Changes to JES2 exits

Some exit routines may have to be changed in order to support the changes to JES2 in z/OS V1R9.

### 21.5.1 \$JCT eye catcher

One change that affects most exits is a change to the JES2 JCT eye catcher field name in the \$JCT mapping macro. Prior to z/OS V1R9, the \$JCT eye catcher field name was JCTID. In z/OS V1R9, the \$JCT eye catcher field name is changes to JCTIDENT. This change is done to resolve the conflict between the JES \$JCT mapping macro eye catcher and the MVS JCT mapping macro which also has an eye catcher field named JCTID.

If your exit routines refer to the JCTID field in the \$JCT, you have to change them to refer to the new field name JCTIDENT.

### 21.5.2 Exit 8 - User environment \$CBIO

Prior to z/OS V1R9, JES2 exit 8 used the MTTR parameter to specify a track address. Starting with z/OS V1R9, JES2 exit 8 uses the MQTR parameter instead. The MQTR parameter was introduced in z/OS V1R7 for large spool volumes. The use of MQTR will eventually replace the use of MTTR.

If you have exit routines that examine field CBMTTR, change them to examine field CBMQTR instead of field CBMTTR.

### 21.5.3 Exit 31 - Allocation SSI

Prior to z/OS V1R9, there was no mapping macro for the exit 31 parameter list. There was simply an area pointed to by a register, with the contents at all applicable offsets described in JES2 documentation. Starting with z/OS V1R9 JES2, exit 31 uses the \$XPL data area instead.

If you use exit 31, you will have to change your exit routine. The \$XPL is passed in R0 and contains the same information as the old parameter list. R1 still points to an area that maps the same as the old parameter list, except for the *condition* and *response bytes*. These are no longer at offsets +1 and +2 in the area pointed to by R1. They are only in the \$XPL.

### 21.5.4 Exit 42 and exit 45

As described under 21.2, “SSI requests authorization enhancements” on page 352, the notify user and SWB modify SSIs in z/OS V1R9 allow unauthorized callers. The SSOB extensions they pass are now potentially in a user key. The storage the SSOB extensions are allocated in must be referenced and updated using the caller’s key. To help exit writers, the fields in the SSOB and extensions are now passed individually to the exit in the \$XPL.

When writing exits 42 or 45, there are now two ways to reference fields in the SSOB extensions passed to the exits, either change the exits to use keyed machine instructions to process the SSOB extensions, or use the new individual fields in the \$XPL. The pointer to the SSOB extensions in the \$XPLs for these two exits is renamed. This is done to cause assembly errors if there is an unmodified reference to the SSOB extensions.

## 21.6 JES3 enhancements

In this section we describe the enhancements introduced in z/OS V1R9 JES3. The enhancements to z/OS V1R9 JES3 aim to allow jobs to run longer without having to be recycled due to internal JES3 spool record management limitations.

### 21.6.1 Relief of the OSE buffer number limit

Prior to z/OS V1R9, the JES3 output service was limited by its original design to 2 bytes for the Output Scheduling Element (OSE) buffers for each job. On top of that, the JES3 code that worked with the buffer numbers did so with signed instructions. As a result, the JES3 output service had a limit of 32 K OSE buffers per job. In January 2003, APAR OW55574 introduced changes to JES3 to process the 2 bytes fields with unsigned instructions. This reduced the problem by allowing for 64 K OSE buffers per job.

Long-running applications which allocate a lot of SYSOUT data sets, such as printing applications, may reach the 64 K limit. When JES3 reaches the maximum number of buffers for the OSE chain of a job, it issues the following message:

```
IAT6718 OSE BUFFER LIMIT REACHED
```

When the OSE buffers limit is reached, JES3 ABENDs the job with an S1FB completion code and reason code 0000006E.

In z/OS V1R9 JES3, all the 2 bytes OSE buffer numbers are extended to 4 bytes. This makes it possible for a job to have 64 K times as many buffers as it had before z/OS V1R9. The support for extended OSE buffer numbers is enabled by default when you IPL your JES3 global on a z/OS V1R9 system. It is not required to read the initialization deck to take this default. After the support for extended OSE buffer numbers is enabled, jobs are able to create OSE buffers after the old limit is reached.

To enable fallback to previous versions of JES3 and for migration purposes, a new keyword is provided on the OUTSERV initialization statement to control the creation of extended OSE buffer numbers beyond the old limit. The new keyword is EXTOSENUM= and it takes the values of YES or NO. The default is YES. Specifying EXTOSENUM=NO requires a hot start with refresh.

### 21.6.2 Coexistence considerations

When the JES3 global is running on a z/OS V1R9 system, extended OSE buffer numbers support is enabled and there are other systems in the sysplex running JES3 on a lower level release of z/OS, then some restrictions apply:

- ▶ The job's output cannot be processed by a JES3 global on a lower release. A DSI to a processor running a lower release or a fallback to a lower release causes the following message to be issued for each job that exceeds the old limit:

```
IAT7604 JOB jobname(jobid) EXCEEDS OSE BUFFER NUMBER LIMIT, JOB REMOVED FROM  
OUTPUT SERVICE
```

In this case, you must dump and restore the job using the dump job (DJ) facility before its output can be processed again by the JES3 global.

- ▶ The output for the job that was created after the limit was exceeded cannot be processed by a process SYSOUT (PSO) application that is running on a JES3 local at an earlier release. Examples of PSO applications include the IBM-supplied external writer, the TSO

OUTPUT command, and some vendor printing products. Note that this limitation does not apply to applications using the SYSOUT application programming interface (SAPI).

The fallback support is provided by APAR OA16731. The APAR allows earlier releases of JES3 to use the dump job (DJ) facility to dump a job that exceeded the old OSE buffers limit to tape and then restore it back to the spool. When the job is restored, JES3 renumbers the OSE buffer records and removes any gaps. This allows the job's output to be processed on a JES3 release earlier than z/OS V1R9. Keep in mind that in the unlikely situation that there are no gaps in the job's OSE buffer records, the job cannot be restored by DJ on an earlier release.

### 21.6.3 More efficient use of spool space

Prior to z/OS V1R9, when a spool data set was spun off and printed, the spool space for the job was not reclaimed. Additional spool data sets allocated by the job could not reuse the space of spun off and printed spool data sets, and would rather use more spool space until the job is ended and purged. Long-running jobs would sometimes need to be recycled in order to reclaim the spool space they no longer use.

Starting with z/OS V1R9, JES3 uses spool space for OSEs more efficiently. This results in greater reuse of spool space for jobs that spin off many data sets. A new control block is introduced to keep a list of available spool records (ASR). The control block is mapped by macro IATYASR.

JES3 uses the ASR to keep a list of unused OSE and WOSE records for a job. Then, when new OSE or WOSE records are needed for the job, JES3 scans the job's ASR to check if there are any available spool records that can be reused instead of allocating a new record and increasing the spool space allocated to the job.

The ASR is shown when a job is snapped. Using the **\*START,DC** command, you can also request to snap only the job's ASR, as show in Figure 21-5:

```
*S DC OPTION=(SNP=ASR) J=15456
```

Figure 21-5 Using **\*START,DC** to snap a job's ASR

Archived



## IBM Health Checker for z/OS

The IBM Health Checker for z/OS has been enhanced to make it easier to write and view checks. This chapter introduces several new checks and enhancements. The following topics are discussed:

- ▶ System REXX Check support
- ▶ Extended SDSF Health Checker support
- ▶ New checks

## 22.1 System REXX check support

Checks can now be written in REXX and are scheduled to run under System REXX. It is expected that SYSREXX will increase the number of user and third party checks, because REXX is much easier to code than assembler. A REXX exec check consists of REXX language instructions that are interpreted and executed by System REXX. System REXX execs can run in a non-TSO environment or a TSO environment.

IBM Health Checker for z/OS provides REXX functions as interfaces between a check and IBM Health Checker for z/OS and System REXX, as follows:

- HZSLSTART** A REXX interface indicating that the check has started running, this is the interface to the assembler HZSCHECK REQUEST=OPSTART macro.
- HZSLFMSG** To issue a message with check results in your REXX check exec, you must use the HZSLFMSG function. Use the HZSLFMSG function to:
- ▶ Indicate that you want to issue a message with a HZSLFMSG\_REQUEST="CHECKMSG" request.
  - ▶ Indicate the message number you want to issue with a HZSLFMSG\_MESSAGE\_NUMBER=msgnum input variable.
  - ▶ Define the number of variables and the variables themselves for a message with the HZSLFMSG\_INSERT input variable.
  - ▶ The HZSLFMSG\_RC output variable reports the return code for the HZSLFMSG function.
- HZSLSTOP** Invoke HZSLSTOP to indicate that the check has completed an iteration. The check usually invokes this function at the very end of the check exec. This function saves and returns the HZS\_PQE\_CHKWORK variable containing the 2 K work area for the check.

**Note:** The general rules and limitations of System REXX apply to REXX checks.

## 22.2 Defining a REXX check

REXX checks can be added in the HZSPRMxx parmlib member or via the **MODIFY (F)** command using:

- ▶ ADD | ADDREPLACE

The HZSADDCK macro is used by the HZSADDCK dynamic exit routine to add a check to IBM Health Checker for z/OS. Adding a check includes defining default values, the parameters and routines required to run the check. The exit routine and the check routines run in the IBM Health Checker for z/OS address space.

REXX checks use a traditional message table generated by HZSMSGEN. The same message table may be used for a non-REXX check and a REXX check, and no special REXX keywords are required.

REXX checks are defined as TSO(YES/NO), indicating that these execs will run in a non-TSO environment or a TSO environment.

**Note:** There are two REXX exec samples in SYS1.SAMPLIB:

- ▶ HZSSXCHK - Sample REXX exec check routine
- ▶ HZSSMSGT - Sample message input

## ADD | ADDREPLACE

You can specify the following parameters for REXX exec checks in either the ADD | ADDREPLACE HZSPRMxx parmlib member or their equivalents in the HZSADDCK macro.

```
{ADD|ADDREPLACE},
CHECK=(check_owner,check_name),
{ CHECKROUTINE=routinename
  | EXEC=execname
  ,REXXHLQ(hlq) [,REXXTIMELIMIT=timelimitvalue]
  { [,REXXTSO=YES]
    | [,REXXTSO=NO
      [,REXXIN={NO | YES} ]
  }
}
```

Figure 22-1 New ADD|ADDREPLACE parameters for check

The following are brief descriptions of the parameters for an ADD | ADDREPLACE check:

- CHECKROUTINE** This required parameter specifies a module name for the check you are adding or replacing. The system gives control to the entry point of this module to run the check. The check routine module must be in an APF-authorized library.
- EXEC** This parameter, required for a REXX exec check, specifies the name of the REXX exec containing the check or checks. The REXX exec must reside in SYS1.SAXREXEC.

**Note:** Health Checker does not check for the existence of the specified REXX check.

- REXXHLQ** This parameter, optional for a REXX exec check, specifies the high level qualifier for any input or output data set for the check.
- REXXTIMELIMIT** This parameter specifies the number of seconds that the exec can run under the AXREXX service before it is canceled. The default is forever.
- REXXTSO=YESINO** This parameter, required for a REXX exec check, specifies whether the check runs in a TSO environment or a non-TSO environment. The default is REXXTSO(YES).
- REXXIN=YESINO** This parameter, optional for a REXX exec check, specifies whether or not a non-TSO check requires an sequential input data set. The name of the REXXIN data set will consist of the high level qualifier specified in the HLQ parameter, the exec name specified in the EXEC parameter, and an optional entry code specified in the ENTRYCODE parameter.

**Note:** You can only specify REXXIN(YES) if you specify REXXTSO(NO). REXX execs that are compiled should use the %TESTHALT compiler option to ensure that halt interpretation is honored if REXXTIMELIMIT is used.

All other keywords, not shown in Figure 22-1 on page 367, are applicable to both REXX and non-REXX checks.

### HZSADDCK macro

The HZSADDCK macro is used by the HZSADDCK dynamic exit routine to add a check to the IBM Health Checker for z/OS. Adding a check includes defining default values, the parameters and routines required to run the check.

```
HZSADDCK ...
{ REMOTE=NO ...
  |REMOTE=YES ...
    [{ REXX=NO ...
      |REXX=YES
        ,EXEC=xexec
        [,ENTRYCODE=.xentrycode]
        [,TIMELIMIT=.xtimelimit]
        ,REXXHLQ=xrexh]q
        ,{ REXXTSO=YES
          |REXTSO=NO
            ,{ REXXIN=NO
              |REXXIN=YES ...
```

Figure 22-2 HZSADDCK parameters

**Note:** The HZSADDCK macro has the same “rules” as the REXX checks that are added with ADD | ADDREPLACE CHECK.

**REMOTE=NO | YES** This is an optional parameter that identifies the location of the check. The default is REMOTE=NO, which indicates that the check runs locally in the address space of IBM Health Checker for z/OS. REMOTE=YES indicates that the check runs remotely. Specify REMOTE=YES to indicate that the HZSADDCK macro call comes from a remote check routine.

## 22.2.1 REXX check structure

The following REXX variables are set when IBM Health Checker for z/OS calls a REXX check.

- ▶ HZS\_Handle is a REXX variable the check should not touch. It is implicit input to check functions.
- ▶ HZS\_PQE\_Entry\_Code is the entry code specified when the check was added.
- ▶ HZS\_PQE\_Function\_Code is one of the following:
  - INITRUN is invoked once for the life of the check which lasts until the check is deleted or deactivated.
  - RUN is called to indicate that the remote check should run and do check cleanup after the initial run (INITRUN) of the check, at the specified interval.
- ▶ HZS\_PQE\_CHKWORK is a 2 k work area that is initialized to nulls. Its contents are saved when HZSLSTOP is called. The saved value is available the next time the check is called and calls HZSLSTRT.



The following are new IBM Health Checker for z/OS callable services for System REXX checks:

- ▶ HZSLSTRT is used to indicate that the check has started.
- ▶ HZSLFMSG is used to issue check messages.
- ▶ HZSLSTOP is used to indicate that a check has completed.

## 22.2.2 DEBUG mode

You can turn on debug mode for a check that is not running properly by using the `DEBUG` parameter in the `MODIFY hzsproc` command, in `HZSPRMxx`, or by overtyping the `DEBUG` field in SDSF to `ON`. Running in debug mode can help you debug your check, because in debug mode:

- ▶ Each message line is prefaced by a message ID, which can be helpful in pinpointing the problem. For example, report messages are not prefaced by message IDs unless a check is running in debug mode.
- ▶ Debug messages, which may contain information about the error, are issued only when the check is in debug mode.

**Important:** Make sure that the `REXXOUTDSN` is enabled when the check is `DEBUG(ON)`.

All REXX output (`SAY`, `TRACE`) is saved to the `REXXOUTDSN`. If `REXXOUTDSN` is not enabled, REXX output is lost.

## 22.2.3 Scheduling a REXX check

The check routine will do the following:

- ▶ Check if the System REXX address space (`AXR`) is up and accepting new requests.
  - The Health Checker for z/OS will monitor System REXX ENFs event code 67.
  - All active and enabled REXX checks are run when System REXX availability toggles from unavailable to available.
- ▶ The Health Checker for z/OS uses the assembler macro interface (`AXREXX`) to queue the exec to System REXX.
- ▶ The `AXREXX` completion exit (`HZSAXRCE`) will then:
  - Test to see if `HZSLSTRT` was issued.
  - Issue information message(s) to check message buffer.
  - Issue `HZSCHECK REQUEST=OPCOMPLETE`.
- ▶ `HZSTKCMD` will do “complete” processing when a REXX check is running or scheduled, and the matching System REXX `AXR` address space goes down.

## 22.2.4 DELETE FORCE=YES

The `AXREXX` cancel service uses the “Request Token” returned when a check is scheduled. It is currently a “program only” interface. There is no `AXR` modify command to cancel a check. The `AXR` Request Token is displayed for a running or scheduled REXX check on `DISPLAY CHECK(owner , name) ,DIAG`.

## 22.2.5 Procedure to implement a REXX check

Start off by using the samples provided in the SYS1.SAMPLIB data set and follow these simple steps:

- ▶ Compile REXX sample HZSSXCHK into SYS1.SAXREXEC using the REXXC procedure in CPAC.PROCLIB.
- ▶ Thereafter, run job HZSMSGNJ in SYS1.SAMPLIB, which invokes the REXX procedure HZSMSGEN to build input assembler code for the message table HZSSMSGT.
- ▶ Run an assembler compile and link (ASMLKED) of HZSSMSGT from the library 'userid.TEMP.ASM' which was created by the REXX HZSMSGEN, into a linklist data set that is APF authorized.
- ▶ Add the statements shown in Figure 22-3 to the HZSPRMxx member and bounce the HZSPROC started task.

```
ADD      ,CHECK(IBMSAMPLE,HZS_SAMPLE_REXXIN_CHECK)
        ,EXEC(HZSSXCHK)
        ,HLQ(IBMUSER)
        ,REXTSO(NO)
        ,REXXIN(YES)
        ,MSGTBL(HZSSMSGT)
        ,ENTRYCODE(1)
        ,USS(NO)
        ,VERBOSE(NO)
        ,PARMS('LIMIT(047)')
        ,SEVERITY(LOW)
        ,INTERVAL(ONETIME)
        ,DATE(20070919)
        ,REASON('A SAMPLE CHECK TO DEMONSTRATE AN '
                'EXEC CHECK USING REXXIN.')
```

Figure 22-3 ADD a REXX check to a parmlib member

- ▶ You will get the following output displayed in Figure 22-4 when HZSPROC comes up.

```
HZS0001I CHECK(IBMSAMPLE,HZS_SAMPLE_REXXIN_CHECK): 675
HZSH0011E There are 0000000A (decimal 10) remaining available special
widgets. This is below the limit.
```

Figure 22-4 Health Checker display in the SYSLOG

- ▶ On completion, do a checks (CK) display in SDSF to see your newly added REXX check, as shown in Figure 22-5 on page 371.

**Note:** By following this procedure you should be able to write your own REXX checks (with messages) and add it to the Health Checker.

### Adding a new check

If you have a check that you have defined with parameters and it does not accept the parameters, you can clear the parameter error by issuing the following command to update the check with a null parameter string:

```
F hzsproc,UPDATE,CHECK=(checkowner,checkname),PARM()
```

## Deleting the check

After defining a System REXX check in the HZSPRMxx parmlib member using the F **hzsproc,ADD | ADDREPLACE,CHECK** command, the check is deleted using the **DELETE** command. Now, you want to bring it back again, so the **ADD, CHECK** command is issued. The command fails, with a message specifying the check already exists, even though it does not appear in SDSF or in a display output. This is because when deleting the check, the check definition is still there in the HZSPRMxx parmlib member, and is still loaded in the system.

To correct this situation, and to get the check to run again, issue an **ADDREPLACE,CHECK** statement containing the check definition into a parmlib member, and issue the F **hzsproc,ADD,PARMLIB=** command; the check is now ready to run.

## 22.3 Extended SDSF CK support

There are five new columns (Einterval, Execname, Locale, Verbose and Origin) added to the SDSF Health Checker CK display panel, as shown in Figure 22-5.

**Note:** Only the Einterval and Verbose fields have values that you can overtype.

Display Filter View Print Options Help						
SDSF HEALTH CHECKER DISPLAY SC63				LINE 27-53 (97)		
COMMAND INPUT ==> _				SCROLL ==> HALF		
NP	NAME	EInterval	ExecName	Locale	Origin	Verbo
	CSVTAM_VIT_OPT_ALL	SYSTEM		HZSPROC	HZSADDCK	NO
	CSVTAM_VIT_OPT_PSSSMS	SYSTEM		HZSPROC	HZSADDCK	NO
	CSVTAM_VIT_SIZE	SYSTEM		HZSPROC	HZSADDCK	NO
	GRS_CONVERT_RESERVES	SYSTEM				NO
	GRS_EXIT_PERFORMANCE	SYSTEM		HZSPROC	HZSADDCK	NO
	GRS_GRSQ_SETTING	SYSTEM		HZSPROC	HZSADDCK	NO
	GRS_MODE	SYSTEM				NO
	GRS_RNL_IGNORED_CONV	SYSTEM				NO
	GRS_SYNCHRES	SYSTEM		HZSPROC	HZSADDCK	NO
	HZS_SAMPLE_REXXIN_CHECK	SYSTEM	HZSSXCHK	REXX	HZSPRMPR	NO
	IEA_ASIDS	SYSTEM		HZSPROC	HZSADDCK	NO
	IEA_LXS	SYSTEM		HZSPROC	HZSADDCK	NO
	IXGLOGR_ENTRYTHRESHOLD	SYSTEM		HZSPROC	HZSADDCK	NO
	IXGLOGR_STAGINGDSFULL	SYSTEM		HZSPROC	HZSADDCK	NO
	IXGLOGR_STRUCTUREFULL	SYSTEM		HZSPROC	HZSADDCK	NO

Figure 22-5 SDSF CK display showing the new columns for REXX execs

The new columns in the SDSF CK display are explained as follows:

**EInterval** This is an exception interval that specifies how often the check should run after the check has found an exception. The EInterval={SYSTEM / HALF / hhh:mm} parameter allows you to specify a shorter check interval for checks that have found an exception.

SYSTEM, which is the default, specifies that the EINTERVAL is the same as the INTERVAL.

HALF specifies that the exception interval is defined to be half of the normal interval.

hhh:mm provides a specific exception interval for the check. After raising an exception, the check will run at the exception interval specified. hhh indicates the number of hours, from 0 to 999. mm indicates the number of minutes, from 0 to 59.

**ExecName** This is required for a REXX exec check and specifies the name of the REXX exec containing the check or checks.

<b>Locale</b>	This specifies where the check is running. The value could be HZSPROC, REMOTE or REXX.
<b>Verbose</b>	This specifies the verbose mode desired (either NO or YES). - NO specifies that you do <i>not</i> want to run in verbose mode. - YES specifies that you want to run in verbose mode. Running in verbose mode shows additional messages about non-exception conditions, if the check supports verbose mode.
<b>Origin</b>	This specifies the origin of the check, which will be HSZADDCK, MODIFY, or HZSPRMxx.

## 22.4 New checks available with z/OS V1R9

The following are new checks defined in z/OS V1R9:

- ▶ RACF check
- ▶ UNIX System Services check
- ▶ TSO/E checks
- ▶ Communications Server Health checks
- ▶ PDSE check
- ▶ VSAMRLS checks
- ▶ System Logger checks
- ▶ Contents supervisor checks
- ▶ New supervisor checks

### RACF check

The following new check is available:

**CHECK(IBMRACF,RACF\_SENSITIVE\_RESOURCES)** is an existing check that was updated for z/OS V1R9. Data set name SYS1.SAXREXEC is the exec library for System REXX. The check verifies that the data set is cataloged, and does not have UACC(UPDATE). RACF checks are part of the RACF program product.

### UNIX System Services check

The following new check is available:

**CHECK(IBMUSS,USSPARMLIB)** compares z/OS UNIX System Services current system settings with those specified in the BPXPRMxx parmlib members used during initialization. It verifies the current USS setting to assure a consistent USS environment for the next IPL. All of the parameters are checked.

For the FILESYSTYPE statement, the types specified in the BPXPRMxx parmlib members will be compared to what Physical File Systems are currently running. For the ROOT/MOUNT FILESYSTEM statements, the following will be checked:

- ▶ Mount point
- ▶ Mount mode (RDRW or READ)
- ▶ AUTOMOVE setting
- ▶ Parm(' ')

For the NETWORK statement, only the MAXSOCKETS value is checked for AF\_INET and AF\_INET6.

**Note:** If CHECK(IBMUSS,USSPARMLIB) is deleted, z/OS UNIX must be restarted to re-add the check. This is because it is a remote check.

## TSO/E checks

The following new checks are available:

- ▶ **CHECK(IBMTSOE,TSOE\_USERLOGS)** verifies that USERLOGS are in effect for the SEND command. If USERLOGs are not enabled, contention on the broadcast data set can cause performance problems.
- ▶ **CHECK(IBMTSOE,TSOE\_PARMLIB\_ERROR)** verifies whether there were problems setting the groupings of settings (authorized commands, authorized programs, send settings, and so on) when the IKJTSOxx parmlib members were processed. Syntax errors in IKJTSOxx parmlib members can result in unexpected results in TSO/E processing. Messages are already issued when IKJTSOxx is processed, but may have been lost after the IPL.

**Note:** Neither check has any parameters.

## Communications Server checks

The following new checks are available:

- ▶ **Check(IBMCS,CSTCP\_SYSPLEXMON\_RECOV\_TCPIPStackname)** verifies that GLOBALCONFIG SYSPLEXMONITOR RECOVERY is specified when IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF is configured.
- ▶ **Check(IBMCS,CSVAM\_VIT\_SIZE)** verifies that the VTAM Internal Trace Table (VIT) size is at the maximum value, which provides optimal trace information for problem determination.
- ▶ **Check(IBMCS,CSVAM\_VIT\_OPT\_PSSSMS)** verifies that the VTAM Internal Trace (VIT) PSS and SMS options are active. We suggest that these options always be active for VIT tracing for optimal problem determination.
- ▶ **Check(IBMCS,CSVAM\_VIT\_DSPSIZE)** verifies the size of the VTAM Internal Trace (VIT) dataspace. We suggest that the VIT dataspace size be set to the maximum of 5 to capture sufficient trace entries for optimal problem analysis.
- ▶ **Check(IBMCS,CSVAM\_VIT\_OPT\_ALL)** verifies that not all VTAM Internal Trace (VIT) options are active. When all VIT options are concurrently active, performance might be less than optimal.

**Note:** Checks CSVAM\_VIT\_SIZE, CSVAM\_VITOPT\_PSSSMS, and CSVAM\_VIT\_OPT\_ALL are designed to verify that the correct trace options are set, in order to ensure that first failure data capture does not adversely affect performance.

- ▶ **Check(IBMCS,CSVAM\_T1BUF\_T2BUF\_EE)** verifies that the T1BUF and T2BUF buffer pool allocations are not the defaults when Enterprise Extender is in use.
- ▶ **Check(IBMCS,CSVAM\_T1BUF\_T2BUF\_NOEE)** verifies that T1BUF and T2BUF buffer pool allocations are sufficient for use when Enterprise Extender is not in use.

## PDSE check

The following new check is available:

- ▶ **CHECK(IBMPPDSE,PDSE\_SMSPDSE1)** verifies that the PDSE restartable address space is enabled. If a PDSE-related problem should occur, a system outage could be avoided

with a restart of the SMSPDSE1 address space. The use of the SMSPDSE1 restartable address space is recommended.

### VSAMRLS checks

The following new checks are available:

- ▶ **CHECK(IBMVSAMRLS,VSAMRLS\_DIAG\_CONTENTION)** verifies that there is no VSAMRLS latch contention.
- ▶ **Check(IBMVSAMRLS,VSAMRLS\_SINGLE\_POINT\_FAILURE)** detects and flags single points of failure in the Share Control Data Sets (SHCDS). We recommend that the SCDSs be on unique volumes.

### System Logger checks

The following new checks are available:

- ▶ **CHECK(IBMIXGLOGR,IXGLOGR\_STRUCTUREFULL)** detects any logstreams that have encountered structure full conditions.
- ▶ **CHECK(IBMIXGLOGR,IXGLOGR\_STAGINGDSFULL)** detects any LOGGER Staging datasets that have encountered structure full conditions.
- ▶ **CHECK(IBMIXGLOGR,IXGLOGR\_ENTRYTHRESHOLD)** detects any logstreams that have encountered entry threshold problems.

### Contents supervisor checks

The following new checks are available:

- ▶ **CHECK(IBMCSV,CSV\_LNKLST\_NEWEXTENTS)** verifies that the number of extents used by each data set in the LNKLST has not changed since the LNKLST was activated.
- ▶ **CHECK(IBMCSV,CSV\_LNKLST\_SPACE)** verifies that partition data sets defined in any active LNKLST are allocated with only primary space.
- ▶ **CHECK(IBMCSV,CSV\_APF\_EXISTS)** verifies that the data sets described by entries in the APF list are consistent with data sets that exist on the system.

**Note:** APAR OA12777 is applicable for z/OS R1.4 to R1.8.

- ▶ **CHECK(IBMCSV,CSV\_LPA\_CHANGES)** detects changes in LPA from IPL to IPL. LPA modules size or area are checked.

**Note:** This applies to z/OS R1.9 only.

### New supervisor checks

The following new checks are available:

- ▶ **CHECK(IBMSUP,IEA\_ASIDS)** detects abnormal ASID usage, and detects and warns when an IPL may become necessary due to usage trends in ASIDs
- ▶ **CHECK(IBMSUP,IEA\_LXS)** detects abnormal linkage index (LX, ELX®, SYSLX and SYSELX) usage.

## DFSMS enhancements

Data Facility Storage Management Subsystem (DFSMS) is an integral component of z/OS system software that automatically manages data from creation to expiration. DFSMS provides allocation control for availability and performance, backup/recovery and disaster recovery services, space management, tape management, and reporting and simulation for performance and configuration tuning. Today's z/OS customers want to extend, leverage, and modernize their system environment while protecting their investments, and they require improved management of their infrastructure and applications. In addition, z/OS must enable customers platform for growth and remove growth related constraints and inhibitors for both horizontal (parallel sysplex) and vertical (single system image) growth.

In this chapter we describe the functional enhancements and new functions that are available in DFSMS. The changes we introduce include the following:

- ▶ Basic Access Methods (BAM) performance
- ▶ VSAM System Managed Buffering (SMB) enhancements
- ▶ Allocate Multi-Volume Data Set in the same Storage Facility Image (SFI)
- ▶ Object Access Method (OAM) enhancements
- ▶ DFSMSshsm enhancements
- ▶ DFSMSrmm enhancements
- ▶ Network File System (NFS) enhancements

## 23.1 Basic Access Methods (BAM) performance

In z/OS V1R9, you can use the MULTSDN parameter of the DCBE macro with QSAM. In previous releases, QSAM ignored the MULTSDN parameter. This new support for MULTSDN allows the system to calculate a more efficient default value for the DCB's BUFNO parameter, and reduces the situations where you need to specify a BUFNO value.

z/OS V1R9 DFSMS provides the following enhancements to BSAM and QSAM, as follows:

- ▶ Long-term page fixing for BSAM data buffers with the **FIXED=USER** parameter
- ▶ BSAM and QSAM support for the **MULTACC** parameter
- ▶ QSAM support for the **MULTSDN** parameter

The DCB extension (DCBE) macro must be coded for these enhancements. The format of the DCBE macro is shown in Figure 23-1.

```
[label] DCBE      [ ,BLKSIZE=n]
                  [ ,BLOCKTOKENSIZE={ LARGE | SMALL } ]
                  [ ,CAPACITYMODE=XCAP ]
                  [ ,EODAD=relexp ]
                  [ ,FIXED=USER ]
                  [ ,GETSIZE={ YES | NO } ]
                  [ ,NOVER={ YES | NO } ]
                  [ ,PASTEOD={ YES | NO } ]
                  [ ,RMODE31={ BUFF | NONE } ]
                  [ ,SYNAD=relexp ]
                  [ ,SYNC={ SYSTEM | NONE } ]
```

Figure 23-1 DCBE macro format

Where:

- |                   |  |
|-------------------|--|
| <b>FIXED=USER</b> | With this DCBE option, the caller ensures that the data areas will remain fixed from the time the READ and WRITE macro instruction is issued through the completion of the CHECK or WAIT macro instructions. Failure to keep them fixed may result in a system integrity exposure as the channel program is using the real addresses associated with the data areas. This keyword is new with z/OS V1R9. |
| <b>MULTACC=n</b>  | Allows the system to process BSAM I/O requests more efficiently by not starting I/O until a number of buffers have been presented to BSAM.   |
| <b>MULTSDN=n</b>  | Requests a system-defaulted NCP.   |

### BSAM support

For BSAM in V1R9, if you code a nonzero MULTACC value, OPEN calculates a default number of READ or WRITE requests that you are suggesting the system queue more efficiently. OPEN calculates the number of BLKSIZE-length blocks that can fit within 64 KB, then multiplies that value by the MULTACC value. If the block size exceeds 32 KB, then OPEN uses the MULTACC value without modification (this can happen only if you are using LBI, the large block interface). The system then tries to defer starting I/O requests until you have



issued this number of READ or WRITE requests for the DCB. BSAM will never queue (defer) more READ or WRITE requests than the NCP value set in OPEN.

### QSAM support

For QSAM in V1R9, if you code a nonzero MULTACC value, OPEN calculates a default number of buffers that you are suggesting the system queue more efficiently. OPEN calculates the number of BLKSIZE-length blocks that can fit within 64 KB, then multiplies that value by the MULTACC value. If the block size exceeds 32 KB, then OPEN uses the MULTACC value without modification (this can happen only if you are using LBI, the large block interface). The system then tries to defer starting I/O requests until that number of buffers has been accumulated for the DCB. QSAM will never queue (defer) more buffers than the BUFNO value that is in effect.

## 23.1.1 Long-term page fixing for BSAM data buffers

To improve performance, in z/OS V1R9 BSAM allows certain calling programs to specify that all their BSAM data buffers have been page fixed. This specification frees BSAM from the CPU-time intensive work of fixing and freeing the data buffers itself. The only restriction is:

- ▶ The calling program must be APF authorized, or be in system key or supervisor state.
- ▶ The format of the data set must be either basic format, large format, PDS, or extended format.

**Note:** Compressed format data sets are not supported.

The DCBE macro option “**FIXED=USER**” must be coded to specify that the calling program has done its own page fixing and indicates that the user has page fixed all BSAM data buffers.

A bit in the DCBE control block (DCBEBENEFIX) is set on to indicate that a program can benefit from specifying “**FIXED=USER**” on the DCBE macro. In addition, a bit in the data extent block (DEB2XUPF) is set on to indicate that “**FIXED=USER**” is in effect.

The caller can ensure that the data areas are fixed by doing either of these:

- ▶ Issuing the PGSER FIX macro.
- ▶ Using the GETMAIN or STORAGE macro for a page fixed subpool.

To avoid duplicate page fixing, the user program can test whether it is running on the appropriate release. To do that, the user program can test whether DFARELS is equal to or greater than X'03010900'.

## 23.1.2 BSAM and QSAM support for MULTACC

In z/OS V1R9, the MULTACC parameter of the DCBE macro is expanded, to optimize performance for tape data sets with BSAM, and to support QSAM with optimized performance for both tape and DASD data sets. The calculations used to optimize performance for BSAM with DASD data sets are also enhanced.

When dealing with a tape data set, OPEN supports MULTACC for BSAM and QSAM.

## BSAM support

For BSAM it will work as documented for DASD.

- ▶ If you code a nonzero MULTACC value, OPEN will calculate a default number of read or write requests that you are suggesting the system queue more efficiently.
- ▶ The system will try to defer starting I/O requests until you have issued this many read or write requests for the DCB.

**Note:** BSAM will never **queue** or defer more **read** or **write** requests than the number of channel programs (NCP) value set in OPEN.

## QSAM support

If you code a nonzero MULTACC value, OPEN will calculate a default number of buffers that you are suggesting the system queue more efficiently.

The system will try to defer starting I/O requests until that many buffers have been accumulated for the DCB.

**Note:** QSAM will never queue (defer) more buffers than the BUFNO value that is in effect. IBM recommends setting MULTACC to one half of the MULTSDN value. If you code a MULTACC value that is too large for the system to use, the system ignores the excess amount. However, the absolute upper limit for MULTACC is 255.

### 23.1.3 QSAM support for MULTSDN

In z/OS V1R9, you can use the MULTSDN parameter of the DCBE macro with QSAM. In previous releases, QSAM ignored the MULTSDN parameter. This new support for MULTSDN allows the system to calculate a more efficient default value for the DCB's BUFNO parameter, and reduces the situations where you need to specify a BUFNO value.

The user can use MULTSDN to give a hint to OPEN so it can calculate a better default value for QSAM BUFNO instead of 1, 2 or 5. The user will not have to be dependent on device information such as blocks per track or number of stripes.

QSAM accepts a MULTSDN value for the following data sets:

- ▶ Tape data sets.
- ▶ DASD data sets of the following types:
  - Basic format
  - Large format
  - Extended format (non-compressed)
  - PDS

For these supported data sets types, the system uses MULTSDN to calculate a more efficient value for BUFNO when the following conditions are true:

- ▶ The MULTSDN value is not zero.
- ▶ DCBBUFNO has a value of zero after completion of the DCB OPEN exit routine.
- ▶ The data set block size is available.

When MULTSDN is specified, note that the default number of buffers may be less than what would have been derived without MULTSDN, as shown in Figure 23-2 on page 379.

Data Set Type	DCBBUFNO default without MULTSDN	DCBBUFNO default with MULTSDN
PDSE Member	1	1
Extended format data set in the compressed format	1	1
UNIX file	1	1
Extended format data set (not in the compressed format)	2 * number of stripes * number of blocks per track	MULTSDN * number of stripes * number of blocks per track
Block size equal to or greater than 32 KB (tape)	2	MULTSDN
Block size less than 32 KB (tape)	5	MULTSDN * number of blocks in 64 KB
IBM 2540 card reader or card punch	3	3
PS, PDS	5	MULTSDN * number of blocks per track
Others (including Dummy)	5	5
TSO terminal	5	1

This item decreases the need for the user to set a value for BUFNO.

Figure 23-2 Default buffer numbers for QSAM (with and without MULTSDN)

## 23.2 VSAM system managed buffering (SMB) enhancements

The JCL AMP parameter's SMBVSP keyword lets you limit the amount of virtual buffer space to acquire for direct optimized processing when opening a data set. Before z/OS V1R9, changing that value required editing the JCL statement, which was not practical when running a batch job. In z/OS V1R9, VSAM provides a simpler, more efficient way of modifying the SMBVSP value, by specifying it for a data class using ISMF. The system managed buffering field on ISMF's DATA CLASS DEFINE/ALTER panel lets you specify the value in kilobytes or megabytes, which SMB then uses for any data set defined to that data class. With this method, the effect of modifying the SMBVSP keyword is no longer limited to one single job step, and no longer requires editing individual JCL statements.

In addition, a new JCL AMP keyword, MSG=SMBBIAS, lets you request a message which displays the record access bias that is specified on the ACCBIAS keyword or chosen by SMB in the absence of a user selection. The message, IEC1611, is issued for each data set that is opened. The new keyword is optional and default is to not issue a message. You should avoid the keyword when a large number of data sets are opened in quick succession.

### 23.2.1 SMB overview

System-managed buffering (SMB), a feature of DFSMSdftp, supports batch application processing. SMB uses formulas to calculate the storage and buffer numbers needed for a specific access type. Each algorithm is called an access bias. SMB takes the following actions, as follows:

- ▶ It changes the defaults for processing VSAM data sets. This enables the system to take better advantage of current and future hardware technology.
- ▶ It initiates a buffering technique to improve application performance. The technique is one that the application program does not specify. You can choose or specify any of the four processing techniques that SMB implements:

#### Direct Optimized (DO)

The DO processing technique optimizes for totally random record access. This is appropriate for applications that access records in a data set in totally random order. This

technique overrides the user specification for nonshared resources (NSR) buffering with a local shared resources (LSR) implementation of buffering.

- Sequential Optimized (SO)** The SO technique optimizes processing for record access that is in sequential order. This is appropriate for backup and for applications that read the entire data set or a large percentage of the records in sequential order.
- Direct Weighted (DW)** The majority is direct processing; some is sequential. DW processing provides the minimum read-ahead buffers for sequential retrieval and the maximum index buffers for direct requests.
- Sequential Weighted (SW)** The majority is sequential processing; some is direct. This technique uses read-ahead buffers for sequential requests and provides additional index buffers for direct requests. The read-ahead will not be as large as the amount of data transferred with SO.

## 23.2.2 Installation considerations

Using the new SMB enhancement, you can:

- ▶ Set the storage limit used by SMB DO from the DFSMS data class as shown in Figure 23-3.

```

Panel  Utilities  Scroll  Help
-----
                                DATA CLASS ALTER                                Page 2 of 5
Command ===> _____
SCDS Name . . . . : SYS1.SMS.SCDS
Data Class Name : DJLVSAM2

To ALTER Data Class, Specify:

Data Set Name Type . . . . . EXT_ (EXT, HFS, LIB, PDS, Large or blank)
If Ext . . . . . R (P=Preferred, R=Required or blank)
Extended Addressability . . . Y (Y or N)
Record Access Bias . . . S (S=System, U=User or blank)
Space Constraint Relief . . . N (Y or N)
Reduce Space Up To (%) . . _ (0 to 99 or blank)
Dynamic Volume Count . . . _ (1 to 59 or blank)
Compaction . . . . . Y (Y, N, T, G or blank)
Spanned / Nonspanned . . . _ (S=Spanned, N=Nonspanned or blank)
System Managed Buffering . . 60M_ (1K to 2048M or blank)

Use ENTER to Perform Verification; Use UP/DOWN Command to View other Panels;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.

```

Figure 23-3 SMB value set in the data class

An ISMF list of the data class, showing the resulting SMBVSP value is shown in Figure 23-4 on page 381.

```

Panel List Utilities Scroll Help
-----
                                DATA CLASS LIST
Command ==> _                               Scroll ==> HALF
                                           Entries 1-1 of 1
                                           Data Columns 14-19 of 45

CDS Name : SYS1.SMS.SCDS

Enter Line Operators below:

   LINE      DATACLAS  VOLUME  ADDITIONAL  DYNVOL  SMBVSP  CISIZE  % FREE
   OPERATOR   NAME        COUNT    VOLUME AMT  COUNT   COUNT   DATA   SPACE CA
---(1)----- --(2)---- -(14)-  ---(15)--- -(16)-  --(17)-- --(18)- --(19)--
          DJLVSAM2      59          -----  --          60M      -----  -----
-----
                                BOTTOM OF DATA

```

Figure 23-4 SMBVSP value in data class list

- ▶ Code MSG=SMBBIAS in JCL to request a VSAM open message indicating what access bias SMB actually is used for a particular component being opened, as shown in Figure 23-5.

```

15.00.02 SYSTEM1 JOB00028 IEC161I
001(DW)-255,TESTSMB,STEP2,VSAM0001,,SMB.KSDS,,
IEC161I SYS1.MVSRES.MASTCAT
15.00.02 SYSTEM1 JOB00028 IEC161I 001(0000002B 00000002 00000000
00000000)-255,TESTSMB,STEP2,
IEC161I VSAM0001,,SMB.KSDS,,SYS1.MVSRES.MASTCAT

```

Figure 23-5 AMP MSG=SMBBIAS in JCL

The message format is shown in Figure 23-6.

```

IEC161I 001(Actual Access Bias being used)-255
For Non-D0, extra message indicating buffer numbers:
IEC161I 001(AAAAAAAA BBBB BBBB CCCCCC DDDDDDD)-255
      WHERE AAAAAAAA = HEX VALUE OF BASE BUFND
            BBBB BBBB = HEX VALUE OF BASE BUFNI
            CCCCCC   = HEX VALUE OF PATH BUFND
            DDDDDDD  = HEX VALUE OF PATH BUFNI

```

Figure 23-6 SMB message format

### 23.3 Multi-volume data set in the same storage facility image

The DFSMSdss data set fast replication function requires that all volumes of a data set reside in the same storage facility image (SFI). To take advantage of data set fast replication, DFSMS volume selection is enhanced to prefer candidate volumes that are in the same SFI, when allocating or extending an SMS-managed multi-volume data set that has point-in-time copy volumes requested. The point-in-time copy volumes are requested when the accessibility field in the storage class is set to CONTINUOUS or CONTINUOUS PREFERRED.

To make the best use of the fast replication function, ensure that enough volumes are on one SFI to accommodate the likely increase in allocations to volumes in these storage groups and SFIs.

**Note:** DFSMS will not prefer volumes in the same SFI when no SFIs have a sufficient number of volumes to meet the number of volume required for allocation.

### 23.3.1 Storage facility image (SFI) overview

A storage facility image (SFI) is a storage device image in a physical storage device. For non-LPAR storage device, such as DS6000 or the legacy control unit, each physical box is viewed as an SFI by the software. For newer storage devices that have LPAR capability, such as the DS8000 Model 9A2, each physical box may contain two SFIs.

**Note:** Do not confuse this LPAR capability inside the storage subsystem with the LPAR that z/OS is running on. They have nothing to do with each other!

Each SFI acts like as a separate storage device. Flashcopy requires the source and target volumes to be in the same SFI.

### 23.3.2 DFSMS volume selection enhancement

DFSMS volume selection prefers volumes in the same SFI when allocating or extending a multi-volume data set that has Accessibility=Continuous or Continuous Preferred. Volume selection is based on the four available accessibility parameters as follows:

<b>Continuous</b>	Specifies that DFSMS must select point-in-time copy volumes and reject non-point in time copy volumes.
<b>Continuous Preferred</b>	Specifies that DFSMS prefers point-in-time copy volumes over non-point-in-time copy volumes.
<b>Standard</b>	Specifies that DFSMS prefers non-point-in-time copy volumes over point-in-time copy volumes.
<b>Nopref</b>	Specifies that DFSMS ignores point-in-time copy capability and treats both point-in-time and non-point-in-time copy volumes equally.

### 23.3.3 Storage Facility Image (SFI) attributes

The storage facility attributes are listed as follows:

- ▶ The DFSMS storage group may have volumes on one or multiple SFIs.
- ▶ SFI is considered when allocating a multi-volume data set with Accessibility=Continuous or Continuous Preferred.
- ▶ A new volume preference attribute with medium importance (i.e. less important than threshold, volume status, etc.)
- ▶ Primary volume must meet all volume selection criteria.
  - Volume that does not meet SFI attribute is not a primary volume.
- ▶ Secondary volumes that meet SFI attribute are ranked higher when more important preference attributes are equal.

The SFI attribute is **not** considered in the following cases.

- ▶ Accessibility=Standard or Nopref, where point-in-time copy volume is less preferred or not preferred.

- ▶ Single volume data set.
- ▶ Requested volsers for guaranteed space request are on different SFIs.
- ▶ Data set already exists on different SFIs.
- ▶ No storage groups have sufficient volumes in the same SFI.
- ▶ No storage groups have sufficient unique controllers in the same SFI to meet the stripe count.
- ▶ Space constraint relief processing is in effect.
- ▶ DFSMS issues message IGD17395I to indicate why a multi-volume data set is not allocated in the same SFI.

### 23.3.4 DFSMS volume selection with SFI attribute

When an allocation is eligible for SFI affinity, DFSMS volume selection takes the SFI attribute into consideration on all types of allocations. We will discuss the effect of SFI on each of these allocations.

#### ▶ **Normal allocation**

- Volumes meet SFI criteria if the SFI has a sufficient number of volumes for selection. DFSMS uses all volume preference attributes, including SFI, to rank all candidate volumes.
- DFSMS will randomly select the best volume among all eligible storage groups.
- DFSMS then selects the remaining volumes from the most preferred volume pool that are in the selected storage group and SFI. If this is not possible, DFSMS will randomly select a volume from this highest ranked volume pool.

#### ▶ **Striping allocation**

- Volumes meet SFI criteria if the SFI has a sufficient number of primary volumes to meet the stripe count. Each unique controller can have only one primary volume.
- DFSMS randomly selects a storage group and SFI that has a sufficient number of primary volumes.
- DFSMS then randomly selects primary volumes from the selected SFI until the stripe count is met.

#### ▶ **Best-fit allocation**

- Volumes meet SFI criteria if they are in the same SFI as the highest ranked volume in each storage group.
- DFSMS selects the storage group and SFI that has the most available space.
- DFSMS then selects the highest ranked volume and the volumes that are in the same storage group and SFI.
- Used mainly by DFDSS for restore and copy. The primary space can be allocated on multiple volumes if none of the volumes can provide sufficient space.

**Note: Primary space may span volumes.**

#### ▶ **EOV new volume extend**

- Volumes meet the SFI criteria if they are in the same SFI as the one the data set resides in.

- DFSMS uses all volume preference attributes to rank the candidate volumes. It then randomly selects one of the best volumes from the highest ranked volume pool.

### 23.3.5 Migration considerations

This enhancement allocates or extends multi-volume data sets to the volumes that are in the same SFI. Therefore, the volumes and storage groups that meet SFI requirement will see more multi-volume data sets and may have higher space utilization.

## 23.4 Object access method (OAM) enhancements

Currently, the OAM storage hierarchy consists of three levels (disk, optical and tape). This enhancement offers further granularity of the tape level by creating two sublevels of tape. This effectively expands OAM's storage hierarchy into four levels:

- ▶ disk
- ▶ optical
- ▶ tape sublevel 1 (TSL1)
- ▶ tape sublevel 2 (TSL2)

In addition to enabling the ability to write and read object data directly to and from a given tape sublevel, OAM provides the ability to transition object data within the tape family (for example: from VTS to native tape). An installation will be able to move data freely in and out of all four hierarchy levels via the OSREQ macro and the OAM storage management component (OSMC) functions.

### 23.4.1 Using OAM enhancements

The following topics lists the types of tasks and associated procedures that you must complete to fully use this enhancement.

- ▶ Update DFSMS storage class constructs
- ▶ Defining tape sublevel parameters to OAM
- ▶ Modifying the SETOAMxx keywords
- ▶ Displaying the new OAM tape level settings

#### Update DFSMS Storage Class constructs

Set the new OAM Sublevel parameter in ISMF to modify existing storage class constructs or create new storage class constructs associated with tape sublevels TSL1 and TSL2 as shown in Figure 23-7 on page 385. Pre-existing tape storage classes will default to TSL1. These storage class construct changes may necessitate updates to ACS routines. Each level of the storage hierarchy is associated with SMS storage class (SC) constructs. These constructs were previously defined with the Initial Access Response Seconds (IARS) and the Sustained Data Rate (SDR) keywords in the ISMF storage class definition panels. The tape sublevel support adds a new OAM Sublevel (OSL) parameter to the storage class, to indicate the sublevel associated with that storage class

- ▶ Initial access response seconds (IARS):
  - 0 = DASD
  - 1-9999 = removable media



- ▶ Sustained data rate (SDR):
  - 0-2 = Optical
  - 3-999 = Tapev
- ▶ OAM sublevel (OSL):
  - 1 = OAM sublevel 1 (default)
  - 2 = OAM sublevel 2

**Note:** A storage class defined with IARS=1, SDR=3 and OSL=2 would equate to TSL2.

```

Panel  Utilities  Scroll  Help
-----
                                STORAGE CLASS ALTER                                Page 1 of 2
Command ==> _____
SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Class Name  : SC54TAPE
To ALTER Storage Class, Specify:
  Description ==> STORAGE CLASS FOR TAPE DATA SETS
  ==> _____
Performance Objectives
Direct Millisecond Response . . . . . _____ (1 to 999 or blank)
Direct Bias . . . . . _____ (R, W or blank)
Sequential Millisecond Response . . . . . _____ (1 to 999 or blank)
Sequential Bias . . . . . _____ (R, W or blank)
Initial Access Response Seconds . . . 60 (0 to 9999 or blank)
Sustained Data Rate (MB/sec) . . . . 10 (0 to 999 or blank)
OAM Sublevel . . . . . 2 (1, 2 or blank)
Availability . . . . . N (C, P, S or N)
Accessibility . . . . . N (C, P, S or N)
Backup . . . . . _____ (Y, N or Blank)
Versioning . . . . . _____ (Y, N or Blank)
Use ENTER to Perform Verification; Use DOWN Command to View next Page;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.

```

Figure 23-7 OAM Sublevel Storclas parameter

### Defining tape sublevel parameters to OAM

The OAM component's parmlib member is CBROAMxx, where xx is the unique suffix specified by the use of OAM=xx parameter in the OAM started procedure, the START command for the OAM started procedure, or in the F OAM,RESTART command. This PARMLIB member is used by OAM during the OAM address space initialization process to determine parameters and configuration information to be used while the address space is active.

New keywords, L2TAPEUNITNAME and L2DATACLASS have been added to the SETOAM statement in the CBROAMxx member of parmlib in support of the new tape sublevel function added in this release. These new keywords will provide the installation with the ability to specify a tape sublevel that will be used at the global and storage group level.

L2DATACLASS(name) is optional parameter that specifies the SMS data class to be used when storing objects to TSL2 for object storage groups that do not have their own L2DATACLASS specification on the STORAGEGROUP subparameter of the SETOAM statement. Tape Sublevel is associated with the OAM Sublevel parameter specified in the DFSMS Storage Class construct.

**Note:** There is NO global level OAM default L2DATACLASS.

L2TAPEUNITNAME(unitname) is a required subparameter of the STORAGEGROUP parameter, if using the TSL2 function. Tape Sublevel is associated with the OAM Sublevel parameter specified in the DFSMS Storage Class construct. For unitname, specify the name of a valid MVS esoteric (group of devices defined to a group name) or a generic unit name. Valid generic unit names are:

- 3480**        A base 3480 device
- 3480x**      A3480 device with the IDRC feature, or a base 3490 device
- 3490**        A 3490E device
- 3590-1**     A 3590 device (or a device that emulates a 3590-1)

Figure 23-8 shows an example of a SETOAM statement in the CBROAMxx parmlib member.

```

EDIT      SYS1.PARMLIB(CBROAM00) - 01.51                Columns 00001 00072
Command ==> _____ Scroll ==> HALF
***** Top of Data *****
000011 SETOAM TAPEDISPATCHERDELAY(45)
000012          TAPERECYCLEMODE(MVSSCRATCH)
000040          STORAGEGROUP(GROUP00 TAPEUNITNAME(3590-1) L2TAPEUNITNAME(ATL3))
000060          STORAGEGROUP(OBJBKP TAPEUNITNAME(3590-1))
000100          MAXRECYCLETASKS(2)
000200          STORAGEGROUP(GROUP00 SGMAXRECYCLETASKS(2))
000210          PERCENTVALID(10)

```

Figure 23-8 SETOAM statement in a CBROAMxx parmlib member

### Modifying the SETOAMxx keywords

You can use the F OAM,UPDATE,SETOAM command to add or change the current L2DATACLASS and L2TAPEUNITNAME specifications for a storage group, or to add or change the L2DATACLASS specification at the global level.

This following command updates the L2DATACLASS value of the SETOAM statement in the CBROAMxx parmlib member, with dataclas signifying the data class to be used for scope. Specify UPDATE,SETOAM with the L2DATAACL or L2TAPEUN keyword.

**F OAM,UPDATE,SETOAM,scope,L2DATAACL,dataclas**

This example updates the L2TAPEUNITNAME value of the SETOAM statement in the CBROAMxx PARMLIB member, with unitname signifying the unit name to be used for scope.

**F OAM,UPDATE,SETOAM,scope,L2TAPEUN,unitname**

### OAM operator commands

The following operator commands have been changed to include the new L2TAPEUNITNAME and L2DATACLASS keywords for the SETOAM statement, as shown in Figure 23-9 on page 387.

Operator Command	--- Changed ---		
	New	Syntax	Results
F OAM, UPDATE, SETOAM	No	Yes	Yes
F OAM, DISPLAY, SETOAM	No	No	Yes
F OAM, DISPLAY, VOL	No	No	Yes
F OAM, START, RECYCLE	No	Yes	Yes
D SMS, STORGRP ( <i>grp-name</i> ), DETAIL	No	No	Yes
D SMS, OSMC, TASK ( <i>OSMC-task</i> )	No	No	Yes
F OAM, START, MOVEVOL	No	No	No
F OAM, START, RECOVERY	No	No	No

Figure 23-9 OAM operator commands

### Displaying the new OAM tape level settings

The F OAM, DISPLAY commands show the new tape level 2 settings, if any are in effect at the global and group levels as shown in Figure 23-10.

F OAM, D, SETOAM, GROUP00
CBR1075I GROUP00 value for SGMXTPS is 1
CBR1075I GROUP00 value for SGMXTPR is 1
CBR1075I GROUP00 value for EXPDATE is /
CBR1075I GROUP00 value for TFULLTHR is 0
CBR1075I GROUP00 value for TFULLPER is 100
CBR1075I GROUP00 value for TAPEUNIT is
CBR1075I GROUP00 value for L2TAPEUN is
CBR1075I GROUP00 value for DMWT is 0
CBR1075I GROUP00 value for DATACL is
CBR1075I GROUP00 value for L2DATACL is MHLTAPE
CBR1075I GROUP00 value for TCOMP is N
CBR1075I GROUP00 value for TDRVSTRT is 9999
CBR1075I GROUP00 value for SGMXREC is 1

Figure 23-10 OAM display command

## 23.4.2 Miscellaneous enhancements

A new SUBLEVEL column is added to OAM's TAPEVOL table in DB2 to indicate which tape sublevel (TSL1 or TSL2) each tape volume is associated with. Valid values are 1, 2 and blank which means the following:

- ▶ The volume is associated with TSL1.
- ▶ The volume is associated with TSL2.
- ▶ The volume is not associated with a sublevel. This only applies to OAM scratch or backup volumes.

The ODLOCFL column in OAM's Object Directory Table which indicates on what storage hierarchy the primary copy of a given object resides, was updated with a new indicator "U" for TSL2.

The values for this column are now:

- D** = DASD
- R** = Recalled
- Blank** = Optical
- T** = Tape (TSL1)
- U** = Tape (TSL2)

### SMF (type 85)

SMF type 85 records have been enhanced to reflect information regarding tape sublevel information. Several SMF OAM subtypes are modified to report tape sublevel information as follows:

- Subtype 2 OSREQ STORE
- Subtype 3 OSREQ RETRIEVE
- Subtype 6 OSREQ DELETE
- Subtype 32 OSMC Storage Group Processing
- Subtype 40 Tape RECYCLE Command
- Subtype 78 LCS Write Tape Request
- Subtype 79 LCS Read Tape Request
- Subtype 87 Demount Tape Volume

### 23.4.3 Migration considerations

Regardless of whether or not your installation intends to exploit the new function, you must modify and run the OAM DB2 migration job "CBRSMR19" from SYS1.SAMPLIB to add the new TSL column to the DB2 TAPEVOL table, and prime it with:

- ▶ "1" for grouped volumes and
- ▶ blank for scratch and backup volumes.

If the customer is running in an OAMplex, and is sharing data across systems, the system administrator must ensure all systems are capable of tape sublevel support, prior to enabling the new support. To ensure all systems are capable of tape sublevel support, all systems in the OAMplex must have the OAM, ISMF and SMS modifications for tape sublevel support installed. (OA17812, PTFs UA32908, UA32909, UA32910 for V1R6, V1R7 and V1R8 respectively)

## 23.5 DFSMSHsm enhancements

DFSMSHsm is a licensed program that automatically performs space management and availability management in a storage device hierarchy. DFSMSHsm makes sure that space is available on your Direct Access Storage Device (DASD) volumes so that you can extend your old data sets and allocate new ones. DFSMSHsm also makes sure that backup copies of your data sets are always available if your working copies are lost or corrupted.

The following enhancements are:

- ▶ Abend 878 reduction
- ▶ Functional statistics record (FSR) improvements

- ▶ Return priority (RP) exit changes

### 23.5.1 Abend 878 reduction

DFSMSHsm captures UCBs in storage below the 16MB line because various functions that it invokes require that UCBs be captured below the line. Because DFSMSHsm has the capability to process thousands of volumes, it captures thousands of UCBs below the 16MB line. Sites that are heavy users of DFSMSHsm, especially those that might be running many tasks simultaneously, have experienced 878 abends. These abends are the result of exhausting below the line storage in DFSMSHsm's private area.

In z/OS V1R9, DFSMSHsm will invoke DFSMSdss via its cross-memory application interface for all functions except "Recall". Invoking DFSMSdss in a separate address space will reduce the amount of storage used in the DFSMSHsm address space. DFSMSHsm will request a unique DFSMSdss address space for each function and Host ID.

#### Details of Abend 878 reduction

The address space identifier for each non-fast replication function will be in the format of "ARCnXXXX", where:

- ▶ n is the unique DFSMSHsm Host ID
- ▶ **XXXX** represents the function, as follows:
  - **DUMP** – HSM Dump
  - **REST** – HSM Restore
  - **MIGR** – HSM Migration
  - **BACK** – HSM Backup
  - **RCVR** – HSM Recover
  - **CDSB** – HSM CDS Backup

**ARC1MIGR** is the name of the DFSMSdss address space started for migration tasks on Host 1.

**Note:** The address spaces are started automatically when the functions are invoked on the DFSMSHsm host. The address spaces exist until DFSMSHsm is shut down. The address spaces automatically terminate when DFSMSHsm is shut down.

#### Display the address spaces

To see the spawned address spaces, issue a display command as shown in Figure 23-11 on page 390.

```

D A,DFSMSDSS
IEE115I 13.26.58 2007.131 ACTIVITY 219
  JOBS      M/S      TS USERS      SYSAS      INITS      ACTIVE/MAX VTAM      OAS
00014      00025      00008      00038      00046      00008/00050      00027
  DFSMSDSS ARC3MIGR IEFPROC  NSW *  A=00A8  PER=NO  SMC=000
                                           PGN=N/A  DMN=N/A  AFF=NONE
                                           CT=000.022S  ET=01.05.11
                                           WUID=STC03360  USERID=STC
                                           WKL=STCTASKS  SCL=STC      P=1
                                           RGP=N/A      SRVR=NO  QSC=NO
                                           ADDR SPACE  ASTE=78B1FA00
  DFSMSDSS ARC3CDSB IEFPROC  NSW *  A=0074  PER=NO  SMC=000
                                           PGN=N/A  DMN=N/A  AFF=NONE
                                           CT=000.071S  ET=104.695S
                                           WUID=STC03361  USERID=STC
                                           WKL=STCTASKS  SCL=STC      P=1
                                           RGP=N/A      SRVR=NO  QSC=NO
                                           ADDR SPACE  ASTE=78B20D00

```

Figure 23-11 DFSMSHsm address spaces

**Note:** These address spaces were created when issuing a DFSMSHsm **migrate** and **CDS backup** command on a LPAR whose DFSMSHsm HOST ID is "3".

You can also do a **SDSF** display to see the HSM DFSMSdss address spaces as shown in Figure 23-12.

```

SDSF DA SC70 SC70 PAG 0 CPU/L/Z 10/ 9/ 0 LINE 1-2 (2)
COMMAND INPUT ==> SCROLL ==> CSR
NP  JOBNAME StepName ProcStep JobID Owner C Pos DP Real Paging SIO
    IEEYSAS ARC3CDSB IEFPROC STC03361 STC NS F4 726 0.00 0.00
    IEEYSAS ARC3MIGR IEFPROC STC03360 STC NS F4 717 0.00 0.00

```

Figure 23-12 SDSF display active

## 23.5.2 Functional statistics record (FSR) improvements

The function statistics record (FSR) is a control block that contains statistics for a particular function that is performed on a data set. It is maintained in the DFSMSHsm work space until the data set processing has completed. Upon completion of the function, the record is written to the DFSMSHsm log and accumulated by category into daily statistics records (DSR) and volume statistics records (VSR) in the migration control data set.

Functional statistics records are key to performing problem analysis with DFSMSHsm. This enhancement includes a number of additions to the information recorded in the FSR as follows:

- ▶ Indicate in the FSR when a recall caused a tape takeaway.
- ▶ Record in the recall FSR the number of times a migrated data set had been recycled before it was eventually recalled.
- ▶ Add the recycle source volume volser to the FSR.

- ▶ Provide the CPU time used for a partial release (FSR type 18) and the expiration of a backup version (FSR type 19).
- ▶ Record the number of tracks needed when an error occurred due to not enough ML1 space.
- ▶ The FSRSTAT program was updated to analyze the new FSR field for recall tape takeaway.

### Details of the FSR enhancement

Each FSR contains values for some fields defined in the **ARCFSR** macro. Some fields have multiple use. The same field means different things based on the **FSRTYPE** or function. One new bit called **FSRF\_RECALL\_TAKEAWAY** is added although the size of the FSR record remains the same. DFSMSHsm modules have been changed to calculate the new field and save it in FSR record.

The following FSR tables in Figure 23-13 and Figure 23-14 on page 392 list the FSR field name, type and information provided.

FSR field name	FSR offset	Type	Length	FSRTYPE	Information provided
<b>FSRF_RECALL_TAKEAWAY</b>	<b>222 (DE)</b>	<b>BITSTRING .... ..1</b>		<b>5</b>	<b>Recall caused a tape takeaway</b>
<b>FSR_RECYCLE_COUNTER</b>	<b>191 (BF)</b>	<b>INTEGER</b>	<b>1</b>	<b>5</b>	<b>Number of times the data set being recalled was recycled on ML2</b>
<b>FSR_RECYCLE_COUNTER</b>	<b>191 (BF)</b>	<b>INTEGER</b>	<b>1</b>	<b>12</b>	<b>Number of times the data set has been recycled on ML2 since its last migration</b>

Figure 23-13 FSR table1

FSR field name	FSR offset # ('x')	Type	Length	FSRTYPE	Information provided
FSR_RECYCLE_SOURCE_VOLSER	98 (62)	CHAR-ACTER	6	10 and 12	Source volume for recycle
FSRCPU	180 (B4)	INTEGER	4	18	CPU time used for partial release processing
FSRCPU	180 (B4)	INTEGER	4	19	CPU time used for expire incremental backup version function
FSR_PSQTY	292 (124)	INTEGER	4	1 and 2	The number of tracks still needed when an error occurred due to "NOT ENOUGH SPACE ON ML1 VOLUME"

Figure 23-14 FSR table2

You can use the DFSMSHsm **REPORT** command or SMF records to view the FSR information.

### 23.5.3 Return priority (RP) exit ARCRPEXT changes

The return-priority exit (**ARCRPEXT**) is taken as each delete, recall, or recover request, in the form of a management work element (MWE), is about to be queued on one of DFSMSHsm's functional subtask queues for processing. For a recall request, this exit is invoked by the host initiating the recall.

#### Extent reduction

Extent reduction is part of HSM's primary space management (PSM). The extent threshold is defined by SETSYS MAXEXTENTS(x) command. During PSM, if DFSMSHsm finds a data set that is not eligible for migration (too young) but has exceeded the MAXEXTENTS value; HSM will schedule a migrate and immediate recall just for extent reduction.

The problem was that, DFSMSHsm did not invoke the ACS routines for recall processing and always recalled the dataset back to its original volume.

The ARCRPEXT exit has been enhanced to give you a choice for dataset recall during extent reduction processing. A field has been added to the Exit to enable customers to indicate that ACS routines should be invoked for extent reduction.

#### Enhancement details

A new bit field "EXTRDCTN" is added to the output parameter list for exit ARCRPEXT.

On return from exit ARCRPEXT, this field is set to either "0" or "1".

- ▶ If the bit field is "0", the dataset will be recalled to its original volume.
- ▶ If the bit field is "1", set "MWEFEXT\_SAMEVOL = OFF" is set.
  - The existing algorithm for ACS routines is then followed.

**Note:** The bit field added to the output parameter list is initialized to "0". This is the default.



## 23.6 DFSMSrmm enhancements

With DFSMSrmm, you can manage your removable media as one enterprise-wide library across systems and sysplexes, and manage your installation's tape volumes and the data sets on those volumes. DFSMSrmm also manages the shelves where volumes reside in all locations except in automated tape libraries.

In this chapter we discuss the following enhancements:

- ▶ Task management support
- ▶ Multitasking of utilities
- ▶ Control data set (CDS) serialization
- ▶ CIM provider
- ▶ JCL data set names
- ▶ Shared parmlib support
- ▶ TSO subcommands
- ▶ 3592 MODEL E05 software support

### 23.6.1 Task management support

Any task in the system that requests DFSMSrmm subsystem services and fails, or is interrupted because a TSO-user used Attention (ATTN), or is cancelled by the operator, results in any corresponding long-running subsystem request failing. In addition, there are checkpoints built into long-running requests so that when the requestor ends (such as a job being cancelled), DFSMSrmm processing is interrupted at a safe and convenient point. Long running local tasks are DFSMSrmm subsystem requests that last long enough to be included in the results of a QUERY ACTIVE command, and the task token can be obtained and used.

If the requester is inventory management, the results of the partial processing are available in the MESSAGE file. Long running tasks that support interruption are:

- ▶ EDGHSKP inventory management, VRSEL, DSTORE, EXPROC, RPTEXT, and CATSYNCH. EDGHSKP ends with return code 12 when cancelled.
- ▶ SEARCHxx subcommands. These end with return code 4, reason code 16 when cancelled.
- ▶ EDGINERS when building lists of volumes to process. EDGINERS processing is still attempted even though one or more search requests of the DFSMSrmm control data set may have been cancelled by the operator. Also, cancelling a task that is processing on behalf of EDGINERS does not cause EDGINERS to be cancelled. To cancel EDGINERS processing, you have to cancel the batch job.
- ▶ ADDxx and DELETExx subcommands with COUNT specified. These end with return code 4, reason code 12 when cancelled.

Additional operator controls are provided to enable simpler management of queued and active tasks running in the DFSMSrmm address space. These controls will allow long running inventory management tasks to be stopped, interrupted, and then restarted. This will enable better management either by system automation or by the operator because of operational priorities.

In previous releases, if a task or address space ended while waiting for DFSMSrmm processing to complete, the processing would continue and only when it was completed would DFSMSrmm check and discover that the requester had ended. Figure 23-15 on

page 394 shows the previous DFSMSrmm task management process. When a task or address space ends while it is waiting for DFSMSrmm processing to be complete, the control blocks identifying the requests are updated to reflect that the requester task has ended. It is only when the subsystem request completes that DFSMSrmm checks if the requester is still waiting and only if still waiting is the requester notified.

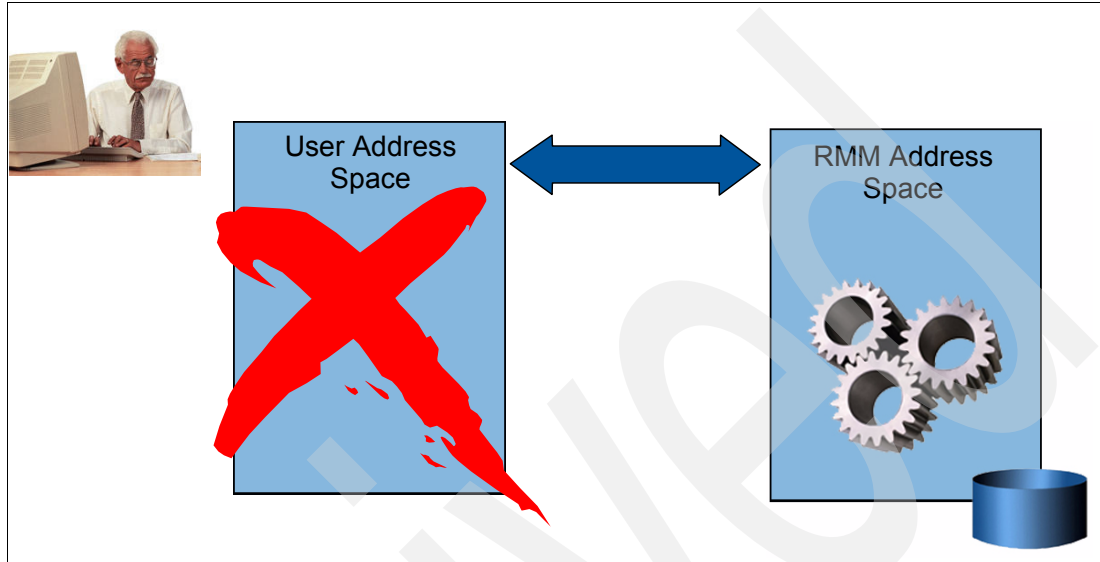


Figure 23-15 DFSMSrmm previous task management process

### New support in z/OS V1R9

In z/OS V1R9, DFSMSrmm long running local requests now check on the requester's status and if the requester has ended, the current processing is interrupted and ended early. For example, if a batch inventory management job is cancelled, DFSMSrmm is notified and inventory management ends early. Also, if a TSO user uses an **attention** (ATTN), DFSMSrmm is notified and any long-running command may be interrupted. DFSMSrmm changed task management process

### Long running tasks

Long running local tasks are DFSMSrmm subsystem requests which last long enough to be included in the DFHSMrmm **QUERY ACTIVE** command display and the task token to be obtained and used. If the requester is inventory management the results of the partial processing will be available in the message file.

Long running tasks which support interruption are:

- ▶ Inventory management
  - VRSEL, DSTORE, EXPROC, RPTEXT, and CATSYNCH
- ▶ **ADD**, **DELETE**, and **SEARCH** TSO subcommands.
- ▶ EDGINERS when building lists of volumes to process.

They all run within the DFRMM address space and check at key points in processing whether they should hold, release or end processing.

**Note:** Task management applies only to long running local tasks.

## Operator commands

In z/OS V1R9, DFSMSrmm operator commands also allow long running requests to be held, cancelled, and released, and allow new requests to be held and released. The **STOP** command processing is changed to prevent DFSMSrmm from stopping if inventory management is running. The operator must now request that inventory management is to end, in order for a **STOP** and **MODIFY** command to process immediately.

Additional operator commands provide ways to manage the running local subsystem requests which can be displayed by **QUERY ACTIVE** operator commands. When **CANCEL**, **HOLD**, or **RELEASE** operator commands are used the command is accepted and the appropriate local tasks marked with the operators request, unless the task is already processing the same operator command. When (ALL/ACTIVE) is used only those local tasks which meet the processing criteria are affected.

The **QUERY ACTIVE** command has been extended to display the subsystem function name rather than by number, and the current task management status as shown in Figure 23-16.

```
F DFRMM,Q A
EDG1119I DFSMSrmm STATUS IS ACTIVE. JOURNAL ENABLED. LISTENER ACTIVE. 152
EDG1120I Function System Task Name Started Token S IP Status
EDG1113I ADD JOB=RMMUSERS 06:15:27 00400009 : :
EDG1113I HSKP JOB=INVMGMTS 05:29:27 00300002H : :
EDG1113I ADD SC70 JOB=RMMUSERS 06:15:49 0060000B+Re<06:17:09
EDG1113I C/S SC70 STC=DFRMM 00:00:00 00700001 Re>06:16:52
EDG1114I LOCAL TASKS 10, ACTIVE 2, SERVER TASKS 2, ACTIVE 2
EDG1122I HELD 1 HELD 0
EDG1118I 1 QUEUED REQUESTS, INCLUDING 0 NOWAIT 0 CATALOG
EDG1123I NEW REQUESTS ARE HELD
EDG1121I DEBUG: DISABLED, PDA TRACE LEVEL: 1-2-3-4- RESERVE:+06:16:45
EDG1101I DFSMSrmm MODIFY COMMAND ACCEPTED
```

Figure 23-16 DFSMSrmm QUERY ACTIVE command

**Note:** Refer to message EDG1113I for a list of Functions and their meaning.

## New operator commands

The following new operator commands have been added:

▶ F DFRMM,CANCEL (token/HSKP/ACTIVE)

The **CANCEL** command option allows the operator to interrupt a long running local task. **HOLD** is used to interrupt a long running local task and cause the task to wait until you are ready to continue. **RELEASE** is used to resume processing When a task has been held.

▶ F DFRMM,HOLD (token/HSKP/ACTIVE/NEW/ALL)

DFSMSrmm will find the first available task running Inventory Management and processes the **CANCEL/HOLD/RELEASE**. You can easily see the first available HSKP task, if any, in the results of the **QUERY ACTIVE** command.

▶ F DFRMM,RELEASE (token/HSKP/ACTIVE/NEW/ALL)

A **CANCEL/HOLD/RELEASE(ALL/NEW/ACTIVE)** allows DFSMSrmm to inform all local and server tasks to process the **CANCEL/HOLD/RELEASE**.

- Using 'HOLD(ALL)' you interrupt all local long running active RMM subsystem request processing and prevent new requests from starting. Using 'RELEASE(ALL)' you

resume all active RMM subsystem request processing and enable new requests to start.

- Using HOLD(NEW) you prevent DFSMSrmm from processing any new, local subsystem requests. Using 'RELEASE(NEW)' you allow DFSMSrmm to processing any new, local subsystem requests. Using 'ACTIVE' you affect only the long running currently active local subsystem requests.
- You should be aware that while you have tasks in HOLD the requester is also in a WAIT and may impact other processing in the system. When you HOLD(ALL) you can release tasks individually via RELEASE(token/HSKP/ACTIVE), but to enable new requests to be processed you must use RELEASE(ALL/NEW).

Figure 23-17 shows the DFSMSrmm messages you will receive after issuing commands:

- ▶ F DFRMM,HOLD(HSKP)
- ▶ F DFRMM,RELEASE(HSKP)
- ▶ F DFRMM,CANCEL(HSKP)

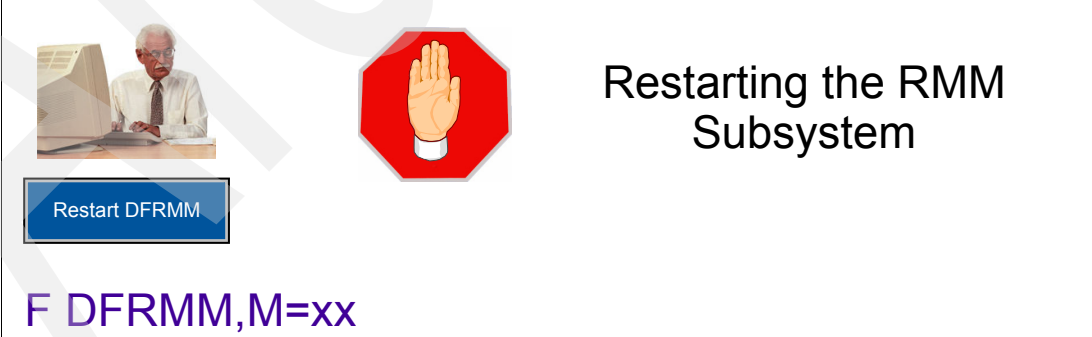
```
EDG2320I PROCESSING HELD BY OPERATOR COMMAND AT 06:43:55
EDG2318I PROCESSING RELEASED BY OPERATOR COMMAND AT 06:44:04
EDG2319I PROCESSING CANCELLED BY OPERATOR COMMAND AT 06:44:12
EDG2303E DFSMSrmm INVENTORY MANAGEMENT TASK ABEND U2223
EDG6901I UTILITY EDGHSKP COMPLETED WITH RETURN CODE 12
```

Figure 23-17 DFSMSrmm messages

### Restarting the DFSMSrmm subsystem

The DFSMSrmm subsystem temporarily stops and reinitializes itself with the new options. Before stopping, DFSMSrmm completes any requests that it is processing. New and queued requests are not processed until reinitialization is completed. The operator response to restarting DFSMSrmm is shown in Figure 23-18. To restart DFSMSrmm and implement new parmlib options, use the following command:

- ▶ F DFRMM,M=xx



Restarting the RMM Subsystem

**F DFRMM,M=xx**

Figure 23-18 DFSMSrmm restart command

### Stopping the DFSMSrmm subsystem

Before you can shut down the DFSMSrmm subsystem, you must wait until all current requests are completed and any outstanding requests are flushed from the request queues. Also, DFSMSrmm cannot stop if inventory management is already running. If any long

running task prevents DFSMSrmm from stopping, you can use this command to interrupt processing:

```
F DFRMM,CANCEL(TaskToken/HSKP)
```

First, issue the **QUERY ACTIVE** command to determine the task that is actually preventing the stopping of DFSMSrmm. If any requests are subject to HOLD processing, you must **RELEASE** or **CANCEL** them in order for DFSMSrmm to **STOP**. If you want to end long running tasks in order to **STOP** DFSMSrmm, issue the **CANCEL** command. You can use this command to display the status of the tasks:

```
F DFRMM,QUERY ACTIVE
```

You must decide whether to cancel the tasks that have been HELD or to release them. To allow the existing tasks to complete while preventing new tasks starting, issue:

```
F DFRMM,HOLD(NEW)
```

Then to release the tasks that are HELD, issue:

```
F DFRMM,RELEASE(ALL)
```

If you decide to cancel the tasks instead, issue:

```
F DFRMM,CANCEL(ALL)
```

In either case, you can now stop the DFSMSrmm subsystem task by issuing:

```
P DFRMM
```

DFSMSrmm will not successfully shutdown if another address space is using its resources. For example, there might be an DFSMSrmm WTOR outstanding for a batch job. If DFSMSrmm shutdown is delayed, DFSMSrmm issues a message to inform you of the delay.

**Note:** DFSMSrmm cannot stop if:

- ▶ Inventory management is already running.
- ▶ If any long running task are in held.
- ▶ If new tasks are in HOLD processing.

## QUIESCE the DFSMSrmm subsystem

To quiesce the DFSMSrmm subsystem use the following command:

- ▶ F DFRMM,QUIESCE

DFSMSrmm completes any requests being processed and then stops all activity. Queued request are not processed until you issue a command to take DFSMSrmm out of the quiesced state and reinitialization is completed. If you stop DFSMSrmm from the quiesced state and any requests are outstanding, message EDG1107D prompts you with your choices of action. The quiesce command together with the operator responses are shown in Figure 23-19 on page 398.

**Note:** DFSMSrmm cannot be quiesced if any long running task are in HELD.

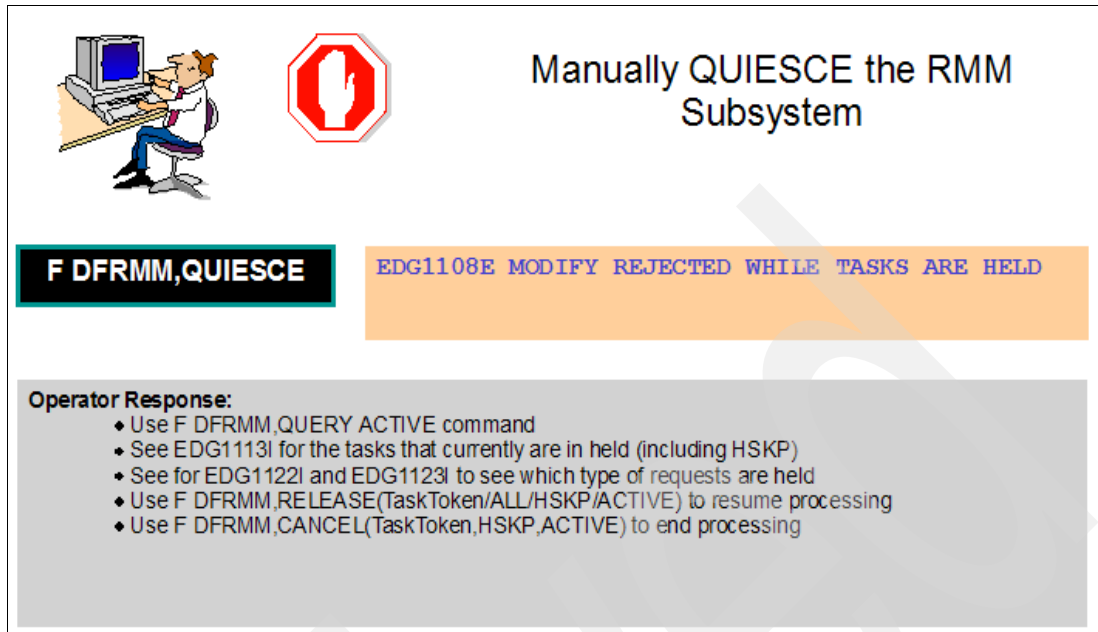


Figure 23-19 DFSMSrmm QUIESCE command

## 23.6.2 Multitasking of utilities

DFSMSrmm utilities, EDGUTIL and EDGHSKP, interaction with system managed volumes in an IBM system managed library is improved through multiple changes that should, especially in larger VTS installations, result in shorter elapsed time and more flexibility. The following are discussed:

- ▶ EDGHSKP EXPROC
- ▶ EDGUTIL
- ▶ EDGSPLCS

### EDGHSKP EXPROC

In z/OS V1R9, DFSMSrmm scratching of system managed volumes can be handled either by writing control statements to an output file to be processed once EXPROC has completed or synchronously as is done today. The trigger is the new processing parameter that can be specified along with the selection options.

When you specify the EXPROC execution parameter an optional SYSIN file allows you to select which subset of the available locations, and volume entries is to be processed during expiration. By default, all volumes in all eligible locations are processed.

Figure 23-20 on page 399 is an example of using EXPROC.

```

//NAID00CN JOB (POK,999),MSGCLASS=T,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=SC63
//HSKP EXEC PGM=EDGHSKP,
// PARM='EXPROC'
//MESSAGE DD DISP=SHR,DSN=NAID00.HSKP.MESSAGES
//EDGSPLCS DD DISP=SHR,DSN=NAID00.SPLCS.DATA
//SYSIN DD *
EXPROC VOLUMES(DVO*,T*,MARY01) EDGSPLCS(YES)

```

Figure 23-20 EXPROC JCL

## EXPROC command

The SYSIN **EXPROC** command does the following:

- ▶ Selects the volumes to be processed.
- ▶ Specifies when the volumes are to be scratched.

The EDGSPLCS DD:

- ▶ Is written during EDGHSKP EXPROC if EDGSPLCS(YES) specified in SYSIN.
- ▶ Instead of scratching the volumes housekeeping creates statements to be used with the EDGSPLCS Utility.

Figure 23-21 shows the format of the EDGHSKP EXPROC SYSIN statement.

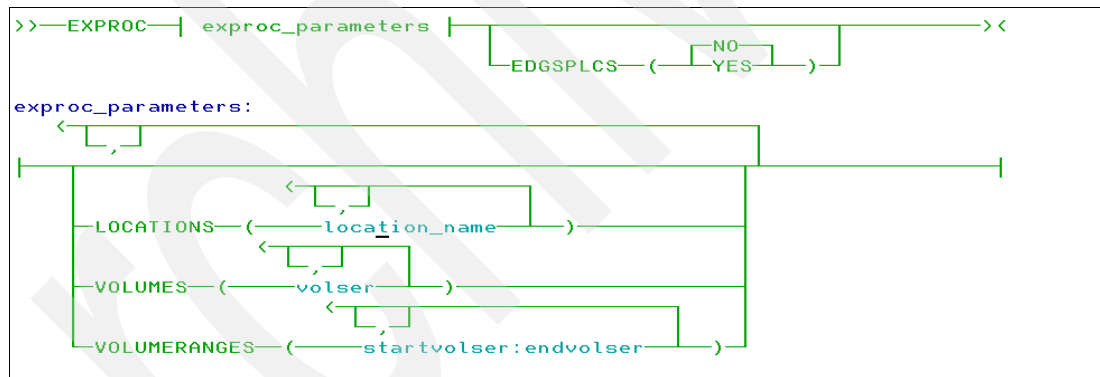


Figure 23-21 EDGHSKP EXPROC - SYSIN statement

Use this command one time only to specify the selection criteria for EXPROC processing. The selection criteria you specify selects volumes for processing by EXPROC which includes the following aspects of expiration processing:

- ▶ releasing expired volumes.
- ▶ setting and processing individual release actions.
- ▶ returning volumes to scratch status.

If you have specified the EXPROC job step parameter but neither VRSEL nor DSTORE Global confirmed actions and moves are not processed.

The operands are all optional and are as follows:

**LOCATIONS** This operand allows you to select a subset of the available volumes, based on the volume current location, for processing.

- VOLUMES** This operand allows you to specify a list of volumes to be processed.
- VOLUMERANGES** Beginning and end of range to be processed.
- EDGSPLCS** Use this operand to request that instead of DFSMSrmm returning system managed volumes to scratch, that statements are generated for use with EDGSPLCS instead.

Figure 23-22 and Figure 23-23 shows an example of the statements in the EDGSPLCS file and MESSAGE file. These statements were generated when we ran the EXPROC job in Figure 23-20 on page 399.

S	TS4285	LIB1
S	DV0042	LIB2
S	DV0053	LIB2
S	DV0061	LIB2

Figure 23-22 EDGSPLCS file

A scratch "S" statement is written to the EDGSPLCS file for each volume for which DFSMSrmm processing would normally have issued a CUA® (change use attribute) request to OAM to return the volume to scratch when EDGSPLCS(NO) is specified.

```

EDG6001I INVENTORY MANAGEMENT STARTING ON 2007/135 AT 09:20:44
          PARAMETERS IN USE ARE DATEFORM(J),EXPROC
EDG6013I THE SYSIN OPTIONS CURRENTLY IN USE ARE
          EXPROC
          EDGSPLCS(YES)
          VOLUMES(DV0*,T*,MARY01)
EDG2309I THE PARMLIB OPTIONS CURRENTLY IN USE ARE
          .....
EDG2308I CHANGES HAVE BEEN MADE TO VRS POLICIES SINCE THE PREVIOUS INVEN
EDG2420I PHYSICAL VOLUMES READ                =          2151
EDG2420I LOGICAL VOLUMES READ                 =           15
EDG2424I TOTAL VOLUMES, THIS RUN, SET PENDING RELEASE =           0
EDG2425I TOTAL VOLUMES RETURNED TO SCRATCH    =           1
EDG2426I TOTAL NUMBER OF SCRATCH RECORDS WRITTEN =           4
EDG2429I MAIN INVENTORY MANAGEMENT UPDATES HAVE COMPLETED SUCCESSFULLY
EDG2307I INVENTORY MANAGEMENT TASK EXPROC   COMPLETED SUCCESSFULLY
EDG6901I UTILITY EDGHSKP COMPLETED WITH RETURN CODE 0

```

Figure 23-23 MESSAGE file

If the run of EDGHSKP includes parameters other than EXPROC:

- ▶ DFSMSrmm processing will process all volumes but only the selected volumes will be subject to EXPROC.
- ▶ Global confirmed actions and moves are completed if DSTORE or VRSEL are specified.



## EDGUTIL

In z/OS V1R9, DFSMSrmm utility EDGUTIL is no longer considered to be part of inventory management processing. EDGUTIL can run at any time, even in parallel with other EDGUTIL instances. The requirement for MEND to be run against an unused CDS is unchanged.

- ▶ A subset of volumes can be selected to be processed on a run of EDGUTIL. An optional SYSIN file allows you to select the subset from the available locations, and volume entries during verification of volumes.

Verification processing of volumes includes:

- ▶ VERIFY/MEND(SMSTAPE/VOLCAT)
- ▶ VERIFY or MEND
- ▶ VERIFY/MEND(ALL)
- ▶ VERIFY/MEND(VOL)

### Processing subsets of volumes

By default, all volumes are verified. The SYSIN commands in Figure 23-24 can be used to select the subset of volumes.

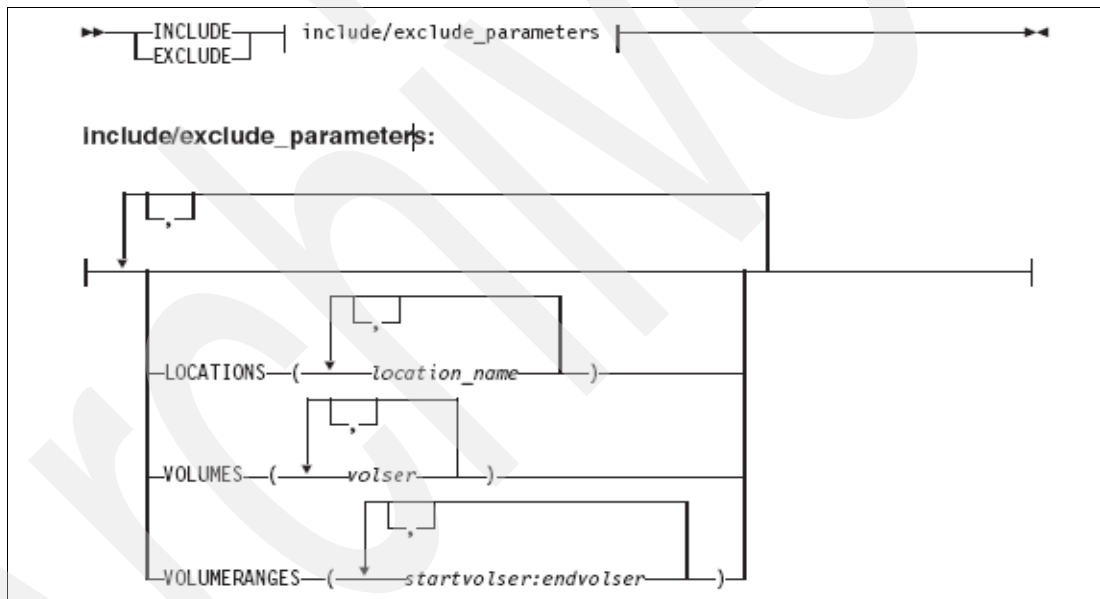


Figure 23-24 EDGUTIL SYSIN statement

You can specify only one INCLUDE and one EXCLUDE statement, in any order. The command operands are all optional.

You can specify one or more of the operands, which are as follows:

#### LOCATIONS

This operand allows you to select a subset of the available volumes based on the volume current location for processing. For 3-way audit and VOLCAT you should specify system managed library location names. For VOL processing any system managed library, or storage location name known to DFSMSrmm, or SHELF can be specified.

#### VOLUMES

This operand allows you to specify a list of volumes to be processed.

#### VOLUMERANGES

Use the VOLUMERANGES operand to select a subset of volumes based on starting and ending volser.

## Processing stacked volumes

In Z/OS V1R9 when a subset of volumes is processed, VERIFY/MEND(VOL) does not consider stacked volume processing. For stacked volumes, VERIFY/MEND(VOL) checks the consistency of exported logical volumes with stacked volumes.

## Deferred correcting TCDB and LM database

Correction of the tape configuration database (TCDB) and the library management database (LM) based on the RMM CDS information was previously done by using EDGUTIL with PARM='MEND(SMSTAPE)'. The correction was done synchronously to the verify processing, resulting in a long elapsed time.

In z/OS V1R9, by using EDGUTIL with PARM='VERIFY(SMSTAPE)' and EDGSPLCS DD and the EDGSPLCS utility:

- ▶ Control statements are generated in the EDGSPLCS output file by EDGUTIL.
- ▶ Volumes can be corrected once EDGUTIL processing is completed by using these statements as an input for the EDGSPLCS utility.

Figure 23-25 is an example of using the deferred TCDB and LM database correction method.

```
//NAID00MN JOB (POK,999),MSGCLASS=T,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=SC63
//HSPK EXEC PGM=EDGUTIL,PARM='VERIFY(SMSTAPE)'
//EDGSPLCS DD DISP=SHR,DSN=NAID00.SPLCS.DATA
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
INCLUDE VOLUMES(M*,V*,T*)
/*
```

Figure 23-25 EDGUTIL VERIFY(SMSTAPE)

**Note:** The EDGSPLCS DD:

- ▶ Is written during VERIFY(SMSTAPE).
- ▶ If specified it causes EDGUTIL to create statements to be used with the EDGSPLCS utility.

Figure 23-26 is the output from SYSPRINT DD.

```
INCLUDE VOLUMES(M*,V*,T*)
EDG6433I STARTING VERIFICATION OF VOLUME RECORDS
EDG6824I VOLUME TST020 IS IN VOLUME CATALOG ERROR STATUS 0004 SECURITY CONFLICT
EDG6846I VOLUME CATALOG UPDATE REQUIRED - STATEMENT WRITTEN TO EDGSPLCS FILE
FOR VOLUME TST020
EDG6824I VOLUME TST021 IS IN VOLUME CATALOG ERROR STATUS 0004 SECURITY CONFLICT
EDG6846I VOLUME CATALOG UPDATE REQUIRED - STATEMENT WRITTEN TO EDGSPLCS FILE
FOR VOLUME TST021
EDG6418W CONTROL DATA SET VERIFY COMPLETED WITH ERRORS
EDG6901I UTILITY EDGUTIL COMPLETED WITH RETURN CODE 4
```

Figure 23-26 SYSPRINT DD output

The explanation of message EDG6846I is:

- ▶ You are running the DFSMSrmm EDGUTIL utility to VERIFY SMS information for volumes. DFSMSrmm found inconsistencies between the DFSMSrmm information for the volume and the volume catalog and Library Manager database. This message indicates that the error can be corrected. A control statement has been written to the data set represented by the EDGSPLCS DD statement.

Figure 23-27 is the corresponding EDGSPLCS generated statements.

S	TST020	LIB1
P	TST021	LIB1

Figure 23-27 EDGSPLCS DD statements

### 3-way audit

For 3-way audit with system managed libraries, VERIFY(SMSTAPE), MEND, and MEND(SMSTAPE) processing exploits the use of a host library interface to return multiple volumes in a single request. In addition, DFSMSrmm allows the selection of libraries and subsets of volumes. This processing reduces the EDGUTIL elapsed time. EDGUTIL constructs a series of requests for the libraries containing the volumes to be verified. The information is retrieved as required and is processed together with entries from the TCDB and volume information from the DFSMSrmm control data set. When a mismatch is detected between the TCDB, DFSMSrmm control data set, and the library manager data, DFSMSrmm uses the CBRXLCS QVR request so that any timing related change can be detected. Processing of all volumes, regardless of function, is subject to the subsetting through location and volume selection.

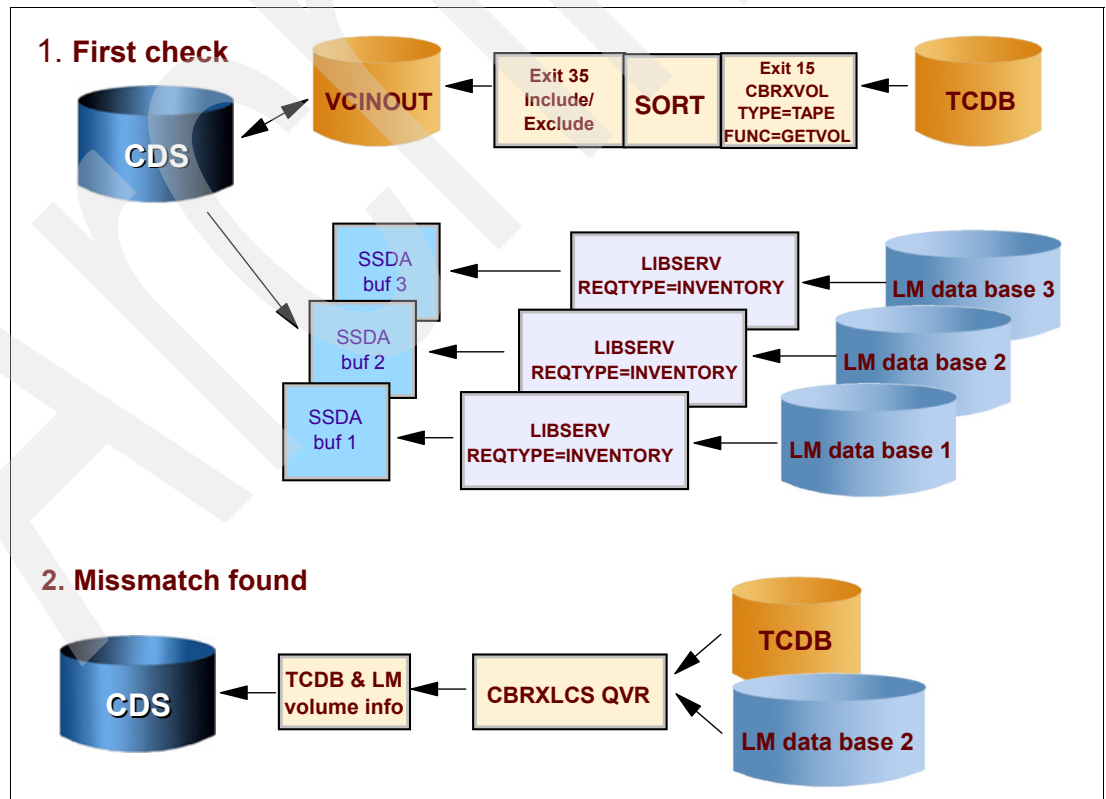


Figure 23-28 EDGUTIL 3-way audit

Using the diagram in Figure 23-28 the following, 1) First check, explains how 3-way audit works:

- ▶ The TCDB volume entries are retrieved from all connected volume catalogs, sorted and the volume subset extracted via the INCLUDE/EXCLUDE statements, and written into the VCINOUT file.
- ▶ The CDS and the VCINOUT are read sequential in parallel, and the data for each volume are compared.
- ▶ If a volume is recorded in VCINOUT (that is in the TCDB) and not in the CDS, or a volume is recorded in the CDS but not in the VCINOUT (that is, it is not in the TCDB), or a mismatch is found between the data in the CDS and TCDB the processing continues at **point 2**.
- ▶ If the volume is not in the buffer, a LIBSERV REQTYPE=INVENTORY is sent to the Library manager of the according library, and 100 records are read at once and saved in buffer. There is 1 buffer for each library. The volume information from the CDS and TCDB are compared to the ones in the buffer.
- ▶ If a volume is recorded in the CDS and TCDB but not in the LM data base, or a mismatch is found between the data in the CDS, TCDB and LM the processing continues at **point 2**.

## 2) Mismatch Found

- ▶ Because the data in VCINOUT (from TCDB) and in the buffered data from LM could be out of date, the LM data base and TCDB are read once more by using the CBRXLCS QVR macro. These data are compared once more with the volume information from the CDS.
- ▶ If the volume is defined to TCDB but not defined to RMM, then it is reported to the MESSAGE file.
- ▶ If the volume is defined to RMM but not defined to TCDB, then it is reported to the MESSAGE file, and if MEND(SMSTAPE) and the Library is an MTL then manual cartridge entry is done.
- ▶ If the volume is defined to RMM and to TCDB but not defined to LM, then it is reported to the MESSAGE file.
- ▶ If a mismatch found then it is reported to the MESSAGE file and if VERIFY(SMSTAPE) and EDGSPLCS DD specified then control statements are written to use with the EDGSPLCS Utility.
- ▶ If MEND then RMM volume records are fixed using the TCDB and LM data.
- ▶ If MEND(SMSTAPE) TCDB and LM volume records are fixed using the RMM data.
- ▶ If MEND(VOLCAT) then the processing is similar, but without LM processing.

## EDGSPLCS

You can use the EDGSPLCS utility to issue supported commands to **OAM** for system-managed volumes. DFSMSrmm builds the input commands for this utility automatically during EDGUTIL VERIFY(SMSTAPE) processing and EDGHSKP EXPROC processing when you request them.

You can run multiple copies of EDGSPLCS. Using different parameters, EDGSPLCS can be processing in parallel for multiple libraries, but this utility does not ensure that each parameter is different from any other currently running. You need RACF ALTER authority to the relevant volume catalog in order to use the EDGSPLCS utility to update the TCDB. For example, if you use just one volume catalog and use the default volume catalog prefix, you need ALTER access to SYS1.VOLCAT.VGENERAL.

Figure 23-29 on page 405 shows the execution parameters for EDGSPLCS.

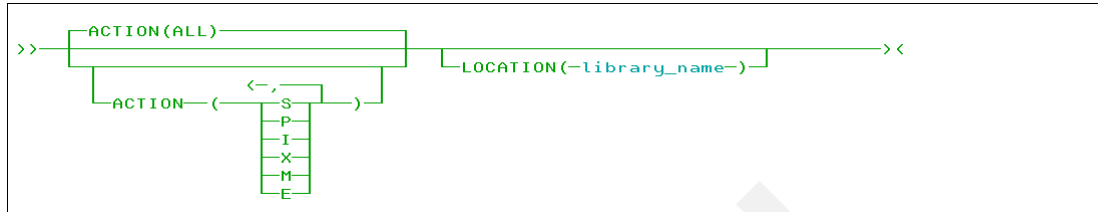


Figure 23-29 EDGSPLCS EXEC parameters

### ACTION(ALL/S/P/I/X/M/E)

- ▶ Specifies that only the specified requests in the input file are processed. You can optionally provide the name of a library to restrict the processing to only those requests. You can specify one or more of the possible actions.
  - S Set to Scratch status.
  - P Set to Private status.
  - I Import volume.
  - X Export volume.
  - M Manual cartridge entry.
  - E Eject volume.

If you do not specify the ACTION parameter, the default value is ALL.

LOCATION(library\_name)

Specifies the name the system-managed library for which the EDGSPLCS utility will process commands during this run. By default, all locations are considered. However, you can select a subset based on the library name using this parameter.

### EDGSPLCS utility example

You can run multiple copies of EDGSPLCS each using different parameters so that processing can be done in parallel for multiple libraries. This is shown in Figure 23-30.

```

//EXEC PGM=EDGSPLCS, PARM=' ACTION (S) , LOCATION (ATLBA999) '
//INDD DD DISP=SHR, DSN=my.edgsplcs.data.set
//OUTDD DD SYSOUT=*

//EXEC PGM=EDGSPLCS, PARM=' ACTION (S) , LOCATION (ATLBA111) '
//INDD DD DISP=SHR, DSN=my.edgsplcs.data.set
//OUTDD DD SYSOUT=*

//EXEC PGM=EDGSPLCS, PARM=' ACTION (P,M) '
//INDD DD DISP=SHR, DSN=my.edgsplcs.data.set
//OUTDD DD SYSOUT=*

```

Figure 23-30 EDGSPLCS utility example

Figure 23-31 shows the output file for each of the JCL EXEC statements run against the input file.

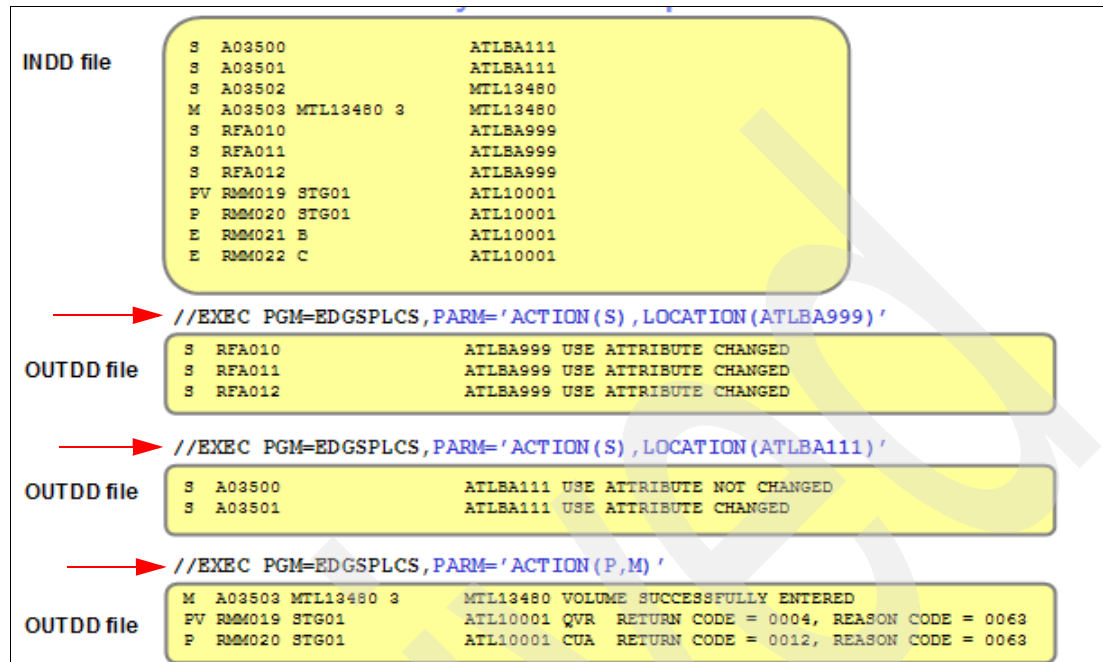


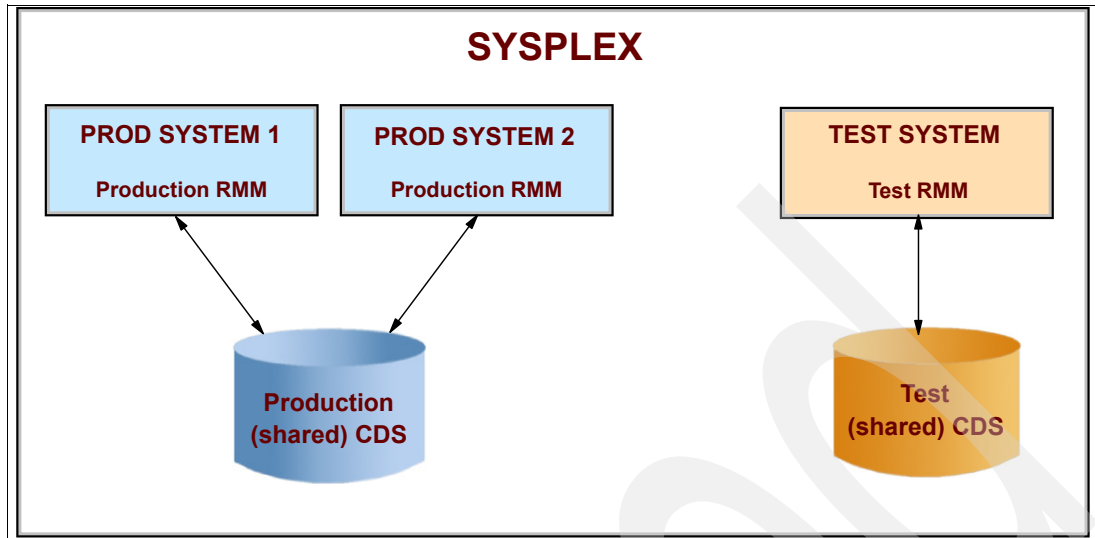
Figure 23-31 EDGSPLCS OUTPUT file

### 23.6.3 Control data set (CDS) serialization

DFSMSrmm now requires you to supply a control dataset identifier in the CSDID parameter of SYS1.PARMLIB member ERBRMMxx, specifying the identifier of the CDS to be used on that system. DFSMSrmm uses the CSDID as part of the ENQ name used to serialize updates to the RMM CDS. The DFSMSrmm CIM Provider also uses the CSDID to distinguish between multiple control data sets. If the CSDID is not yet set in the CDS you can optionally use EDGUTIL UPDATE to set it, otherwise DFSMSrmm will set the CSDID when first started. We recommend that you use a unique CSDID for each CDS.

#### CSD serialization change

The serialization used by DFSMSrmm for its CDS is changed to use a new resource name that includes the CDSID. This avoids conflicts when multiple RMMplexes run in the same sysplex. Lets look at the scenario in Figure 23-32 on page 407.



In previous DFSMSrmm releases, even though the Production and the Test RMM are independent from each other, a write access to the Test CDS would cause an exclusive ENQ of the SYSZRMM MASTER.RESERVE resource. If you use global reserve, the accesses to the Production CDS and to the Test CDS are serialized each after the other. With z/OS V1R9 DFSMSrmm write accesses to the Production CDS and write accesses to the Test CDS are serialized independent from each other.

For example, if your production and test RMM run in the same sysplex, you will get two exclusive SYSTEMS enqueues for write access. One for production and the other for test in parallel as shown in Figure 23-33.

SYSTEM ENQ STATUS	ROW 1 TO 3
MAJOR NAME PREFIX . . . SYSZRMM	(SYSDSN, SPFEDIT, ETC)
MAJOR MINOR	TYPE
SYSZRMM	MASTER.RESERVE.CDSPRODA SYSS
SYSZRMM	MASTER.RESERVE.CDSTEST1 SYSS
SYSZRMM	RMM.ACTIVE SYS

*Figure 23-33 SYSTEM ENQ for write access*

### CDSID parmlib option

There is a new option in the EDGRMMxx parmlib member called CDSID. The format of CDSID is as follows:

**CSDID(ID)** Specifies the identifier of the control data set that must be used on this system. Specify a value one to eight alphanumeric characters long.

Following is the CDSID parmlib option.

- OPTION OPMODE(P) -
- DSNAME(RMM.CONTROL.DSET) -
- JRNLSNAME(RMM.JOURNAL.DSET) -
- CDSID(SC70)** -

## CREATE and UPDATE EDGUTIL parameters

With EDGUTIL when creating a new CDS, the CDSID is set in the CDS control record.

The following JCL statements show how the CDSID is created using EDGUTIL.

```
//EDGUTIL EXEC PGM=EDGUTIL,PARM='CREATE'  
:  
//SYSIN DD *  
CONTROL CDSID(SC70)
```

## Provide a CDSID to an existing RMM system

We recommend that you provide a CDSID prior to the V1R9 installation. Ensure that each control data set has a unique CDSID. By using the RMM LISTCONTROL command, check whether the data element CDSID is filled or not as shown in the following RMM LISTCONTROL command:

```
RMM LISTCONTROL ALL  
System options:  
PARMLIB Suffix = 02  
SMF audit = 0 SMF security = 0 CDS id =
```

If the CDSID field is empty, as is the case in our example, run EDGUTIL with the UPDATE parm as shown in the JCL example that follows and change your parmlib member according:

```
//EDGUTIL EXEC PGM=EDGUTIL,PARM='UPDATE'  
//SYSIN DD *  
CONTROL CDSID(SC70)
```

## DFSMSrmm messages

The following warning message is issued during DFSMSrmm startup.

```
EDG0237E MISSING IDENTIFIER FOR THE CONTROL DATA SET
```

**Explanation:** During initialization DFSMSrmm checks to ensure that you have a CDSID specified in the EDGRMMxx parmlib member. A CDSID is mandatory.

**System Action:** DFSMSrmm initialization stops.

You also get a EDGUTIL message if no CSDID is provided during CREATE and UPDATE SYSIN control statements. EDGUTIL CREATE returns these messages in Figure 23-34 when the CDSID is not provided.

```
EDG6007E IKJ56701I MISSING IDENTIFIER FOR THE CONTROL DATA SET  
EDG6414E CONTROL DATA SET CONTROL RECORD CREATE FAILED  
EDG6901I UTILITY EDGUTIL COMPLETED WITH RETURN CODE 12
```

Figure 23-34 EDGUTIL message for missing CSDID

## Global resource serialization (GRS)

When you change the CSDID, any running systems that share the control data set detects the change in CDSID and changes the ENQ name they use for serialization. Ensure that the GRSRNLxx parmlib member is updated to reflect and CDSID changes you make. RNLDEF must be updated so that each system has a reserve with that systems CDSID specified. This is so that RMM can issue the ENQ on a per system basis.



Depending on how you have determined the GRS parameters, you should also update the GRS resource name list in SYS1.PARMLIB(GRSRNLxx) as shown in Figure 23-35.

RNLDEF RNL (EXCL) TYPE (SPECIFIC)	← remains
QNAME (SYSZRMM)	← remains
RNAME (MASTER.RESERVE)	← remains
RNLDEF RNL (EXCL) TYPE (SPECIFIC)	← to be added for each CDSID
QNAME (SYSZRMM)	← to be added for each CDSID
RNAME (MASTER.RESERVE.cdsid)	← to be added for each CDSID

Figure 23-35 GRS RNLDEF statements

Alternatively the TYPE(SPECIFIC) can be changed to TYPE(GENERIC).

### 23.6.4 Migration and coexistence considerations

When a V1R9 system will ENQ with the new enqueue name, all lower systems in the SYSPLEX, which are using the same shared CDS, must do the ENQ with the same new enqueue name, too. With the coexistence **APAR OA17965** the following functions are installed on lower releases (V1R6 to V1R8):

- ▶ Does the ENQ with the new enqueue name, if the CDSID enqueue setting is enabled.
- ▶ Displays the CDSID enqueue setting with LISTCONTROL, REXX and SFI.
- ▶ A warning message appears if the lower system has not provided the parameter CDSID in PARMLIB.

**Note:** A lower system never switches on the CDSID enqueue setting. This is only be done by a z/OS V1R9 DFSMSrmm system.

### 23.6.5 Common Information Model (CIM) provider

The RMM CIM provider is the link for a customer to obtain real-time RMM data within a Common Information Model (CIM) environment. This functionality will enable a CIM client such as a PC to obtain RMM data.

In z/OS V1R8 DFSMSrmm shipped a Common Information Model (CIM) Provider for use with the OpenPegasus CIMOM. The provider supported a subset of the resources managed by DFSMSrmm. The CIM model supported by DFSMSrmm is further extended in this release to cover all of the resources managed by DFSMSrmm. The CIM provider is updated to support out-of-process mode functionality added to OpenPegasus CIM Server with 2.5.1. This has an impact on the way that the DFSMSrmm provider code is installed and run.

In addition, the RMM extensions to the CIM object model are based on the latest CIM V2.11 specification. To further enable selection of resources via the API the SEARCH subcommands are all now capable of specifying continuation, enabling "chunking" of returned entries.

#### RMM CIM classes

Four new abstract super-classes have been introduced as an intermediate layer between native CIM schema classes and RMM classes. These abstract classes hold a common set of keys for their child classes. Abstract classes can't have instances. Derived child classes can have instances.

Four new abstract super-classes have been introduced as an intermediate layer between native CIM schema classes and RMM classes. These abstract classes hold a common set of keys for their child classes. Abstract classes can't have instances, but derived child classes can have.

The DFSMSrmm CIM interface has been completed by implementing the remaining resources, not covered in V1R8, which are:

- IBMrmm\_Product
- IBMrmm\_PolicyRule
- IBMrmm\_Control

The following CIM classes are new in z/OS V1R9:

- IBMrmm\_LogicalMedia (abstract class)
- IBMrmm\_PhysicalMedia (abstract class)
- IBMrmm\_StorageMediaLocation (abstract class)
- IBMrmm\_Identity (abstract class)
- IBMrmm\_PhysicalVolume
- IBMrmm\_LogicalVolume
- IBMrmm\_Dataset
- IBMrmm\_Owner
- IBMrmm\_Location
- IBMrmm\_ShelfLocation
- IBMrmm\_Product
- IBMrmm\_PolicyRule
- IBMrmm\_Control

### **CIM association classes**

All aspects of a volume to a location or shelf location are now modelled by various new association classes as follows:

- IBMrmm\_PhysicalLogicalVolume (association 1:1)
- IBMrmm\_LogicalVolumeDataset (association 1:N)
- IBMrmm\_LogicalVolumeOwner (association N:1)
- IBMrmm\_DatasetOwner (association)
- IBMrmm\_PhysicalVolumeCurrentLocation (association N:1)
- IBMrmm\_PhysicalVolumeDestinationLocation (association N:1)
- IBMrmm\_PhysicalVolumeHomeLocation (association N:1)
- IBMrmm\_PhysicalVolumeLoanLocation (association N:1)
- IBMrmm\_PhysicalVolumeOldLocation (association N:1)
- IBMrmm\_PhysicalVolumeRequiredLocation (association N:1)
- IBMrmm\_PhysicalVolumeCurrentShelfLocation (association 1:1)
- IBMrmm\_PhysicalVolumeDestinationShelfLocation (association 1:1)
- IBMrmm\_PhysicalVolumeOldShelfLocation (association 1:1)

An example being traversing IBMrmm\_PhysicalVolumeDestinationLocation left-to-right will return the destination location of a volume, which moves because of an inventory management run. Vice versa, when traversing right-to-left, all volumes are listed, that are intended to move into a specific location.

Volume chains and PolicyRule (VRS) chains are also supported with release V1R9 as follows:

- IBMrmm\_LogicalVolumeChainedLogicalVolume (association 1:1)
- IBMrmm\_LogicalVolumeLogicalVolumeInChain (association 1:N)
- IBMrmm\_LocationShelfLocation (association 1:N))
- IBMrmm\_ProductLogicalVolume (association 1:N)

IBMrrmm\_PolicyRuleNextPolicyRule (association N:1)  
 IBMrrmm\_PolicyRuleAndPolicyRule (association N:1)  
 IBMrrmm\_PolicyRulePolicyRuleInChain (association 1:N)  
 IBMrrmm\_PolicyRuleLocation (association N:1)  
 IBMrrmm\_PolicyRuleOwner (association N:1)  
 IBMrrmm\_SearchOperands (aux class for search type operations)  
 IBMrrmm\_DeleteOperands (aux class for delete operation)

The association IBMrrmm\_LogicalVolumeChainedLogicalVolume let you traversing through a chain of volume, back and forth. IBMrrmm\_LogicalVolumeLogicalVolumeInChain returns the whole set for a given chain member. The same is true for IBMrrmm\_PolicyRulePolicyRuleInChain.

IBMrrmm\_PolicyRuleNextPolicyRule let you traverse through normally chained policy rules, while IBMrrmm\_PolicyRuleAndPolicyRule let you traverse through logical policy rules.

**Note:** A policy rule can have more than one previous rule, that's why the N:1 relationship.

### Supported CIM operations

A new method `invokeMethod()` is introduced from the `MethodProvider2` interface as shown in Figure 23-36. Search is the only supported method name for the `invokeMethod` function. It returns a list of objects from `DFSMSrrmm` by name. It is particularly useful when working with the CONTINUE operand for search requests. Only the number of objects that are actually returned to the client are specified by the LIMIT operand. The client is able to get the list incrementally, in an interactive way.

CIM Class	CIM Operations	
Main classes	<code>enumerateInstances</code>	<code>ei</code>
	<code>enumerateInstanceNames</code>	<code>ein</code>
	<code>getInstance</code>	<code>gi</code>
	<code>createInstance</code>	<code>ci</code>
	<code>modifyInstance</code>	<code>mi</code>
	<code>deleteInstance</code>	<code>di</code>
	<code>invokeMethod</code>	<code>im</code>
Association classes	<code>associators</code>	<code>ai</code>
	<code>associatorNames</code>	<code>ain</code>
	<code>references</code>	<code>ri</code>
	<code>referenceNames</code>	<code>rin</code>

Figure 23-36 CIM invokeMethod

### Search operands

Whereas in z/OS V1R8 the `IBMrrmm_SearchOperands` class has no special importance, it gains importance in z/OS V1R9 in conjunction with the CONTINUE operand. The CONTINUE operand is added to the search string within the instances of `IBMrrmm_SearchOperands`.

The CIM search method (`enumerateInstanceNames`) does not allow passing the search operand. Therefore, any operands are set (1) before in the corresponding instance of the class `IBMrrmm_SearchOperands`, which is later used during the search operation (2) as shown in Figure 23-37 on page 412.

To make use of the CONTINUE operand, you have to add it to an instance of IBMmmm\_SearchOperands.

The V1R9 CIM provider fully supports the new CONTINUE operand, to divide a huge set of returned volumes or data sets into multiple sub-sets. You can either specify the CONTINUE operand in the appropriate instance of IBMmmm\_SearchOperands or use the new API method invokeMethod(). Following is an example of listing all volumes from DFSMSrmm is groups of ten, as follows:

- ▶ Point 1 lets the provider incrementally pull chunks of data from RMM, but finally returns the whole matching set to the client, as follows in wbevcli mii:

```
http://<userid>:<password>@<cimon_uri>:<port>
/root/cimv2:IBMmmm_SearchOperands
```

- ▶ Point 2 passes the concept of chunking down to the client, and the client has to keep requesting data until the set is complete and no more data is available, as follows in wbemcil ein:

```
http://<userid>:<password>@<cimon_uri>:<port>
/root/cimv2:IBMmmm_LogicalVolume
```

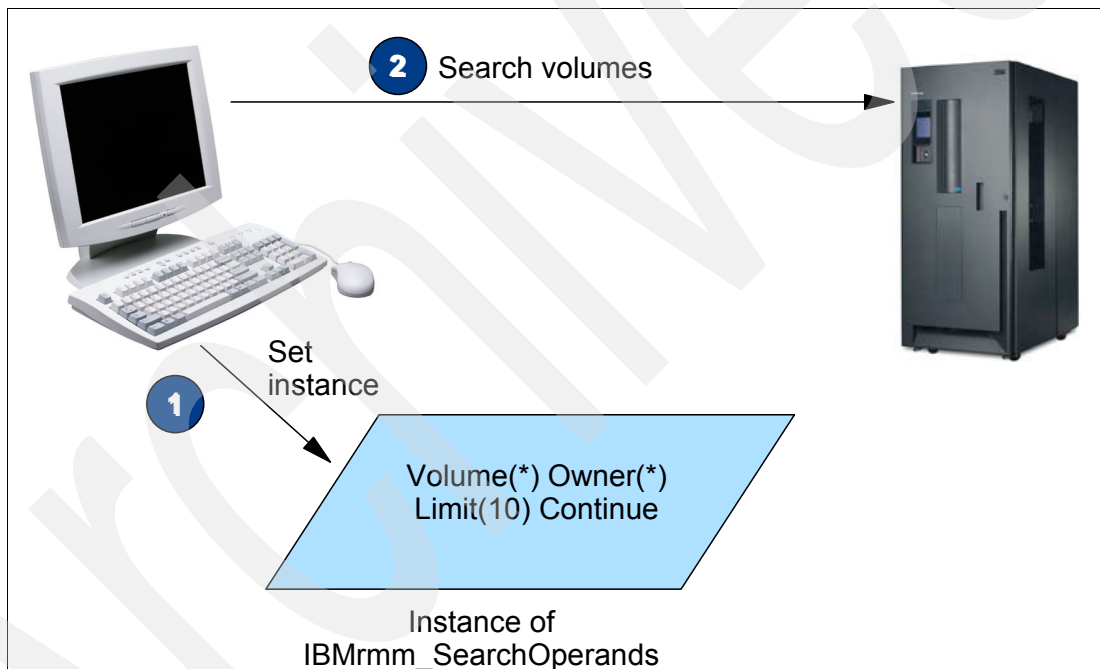


Figure 23-37 CIM search operands

### Software dependencies

A fully functional Pegasus CIM Server is a fundamental prerequisite for the RMM CIM-Provider.

- ▶ For z/OS the Pegasus CIM Server is preinstalled in /usr/lpp/wbem and must be invoked as Unix System Service (USS) process. Refer to the *eServer zSeries Common Information Model User's Guide*, SC33-7998 for how to setup the CIM server, particularly the necessary RACF settings for the CIM server and client user IDs.
- ▶ For LINUX, the provider files has to be downloaded from /usr/lpp/dfsms/rmm to the Linux system. The Pegasus CIM Server 2.5.3 has to be downloaded from the following website:

<http://www.openpegasus.org>

- ▶ The minimum supported Java level is 1.4.2. The designated Java libraries are required to be added to the CLASSPATH. Refer to Figure 23-38 for a list of these libraries. For how to setup the CLASSPATH please refer to the “Exports” sections in the readme file rmmcim.txt.

Library name	Version	Download location
j2ee.jar	1.3.1	<a href="http://java.sun.com/j2ee/index.jsp">http://java.sun.com/j2ee/index.jsp</a>
soap.jar	2.3.1	<a href="http://apache.rmpc.co.uk/ws/soap/version-2.3.1/">http://apache.rmpc.co.uk/ws/soap/version-2.3.1/</a>
mail.jar	1.3.1	<a href="http://java.sun.com/products/javamail/downloads/index.html">http://java.sun.com/products/javamail/downloads/index.html</a>
xerces-2.4.0.jar	2.4.0	<a href="http://www.ibiblio.org/maven/xerces/jars/">http://www.ibiblio.org/maven/xerces/jars/</a>
log4j-1.2.8.jar	1.2.8	<a href="http://logging.apache.org/log4j/docs/download.html">http://logging.apache.org/log4j/docs/download.html</a>
uddi4j.jar	2.0	<a href="http://sourceforge.net/projects/uddi4j">http://sourceforge.net/projects/uddi4j</a>

Figure 23-38 CLASSPATH - Java libraries

### CIM installation

Use script rmmutil.sh or alternatively, load CIM classes and register providers manually by invoking from the shell:

- ▶ `cimmof -l. -nroot/cimv2 rmmcimp.mof`
- ▶ `z/OS: cimmofl -l. -nroot/PG_InterOp -R/var/wbem rmmcimpr.mof`
- ▶ `Linux: cimmofl -l. -nroot/PG_InterOp rmmcimpr.mof`

**Note:** The “Miscellaneous Tests” menu offers various post-installation tests, which we highly recommended you execute, before actually working with the provider. For example the web service or direct API can be tested, to make sure the underlying communication channels work properly.

Set the environmental variables:

- ▶ `export RMMCIM_NAMESPACE=root/cimv2`
- ▶ `export RMMCIM_CONFIG=/var/rmm/rmm.properties`

Set external link (z/OS only):

- ▶ `ln -e EDGXHCLL libEDGXHCLL.so (within $LIBPATH)`

**Note:** The external link is necessary to be able to connect to the direct API C++ DLL EDGXHCLL, which actually resides in SYS1.SIEALNKE.

### Migration and coexistence considerations

IBM WebSphere Application Server (v5.02 or higher) and the z/OS CIM Server V1R9 has to be up and running on the target z/OS system in order to work with the RMM CIM provider V1R9. If running under Linux, OpenPegasus CIM Server 2.5.3 or above has to be installed. The RMM CIM provider readme with setup instructions can be found under `/usr/lpp/dfsms/rmm/rmmcim.txt` within the Unix System Services (USS).

In addition the physical file system (PFS) after the First IPL, requires the following:

The customer has to unregister the V1R8 CIM providers and unload all V1R8 CIM classes, by using the rmmutil.sh tool. Following this, the complete set of V1R9 providers must be registered and the V1R9 CIM classes must be loaded, by using the same tool.

**Note:** Customers migrating from z/OS V1R7 should follow the migration steps documented for migration to V1R8 before completing the V1R9 registration steps.

### 23.6.6 JCL data set names

The data set naming requirements for tape data sets supported by the TSO subcommands and API are relaxed so that any 44 character string can be used. This enables use of unqualified data set names such as those that are accepted by the MVS JCL DSNAME keyword. Quoted data set name strings allow any character string to be specified for a data set name except for leading blank and leading hex zero which are not supported.

**Note:** Quotes make the difference!

The following are the rules for a “quoted” data set name.

- ▶ 'My Data/Set"Name".123!'
  - Can include special characters
  - Can contain lower case characters
  - Can be any string
  - Max 44 characters long
  - Must not start with BLANK

The following are the rules for an “un-quoted” data set name.

- ▶ MVS.DATA.SET.NAME
  - Prefix may be applied
  - Translated to upper case
  - Must follow MVS naming convention

#### RMM subcommands

Data set name validation is only performed when a data set or VRS data set name record might be created in the CDS. This includes:

- ▶ AD dsname
- ▶ CD NEWDSNAME
- ▶ CV DSNAME
- ▶ AV DSNAME
- ▶ AS DSNAME

In all other cases the validation of a data set name is minimal so that any data set recorded or already defined to RMM can be listed changed, searched or deleted.

You can define a retention policy for a specific data set by using a fully qualified data set name in a vital record specification. The vital record specification defined in this example

defines the retention policy for one data set, PRITCHAR.BACKUP.DATA. All copies of the data set should be retained for five days.

```
RMM ADDVRS DSNAME('PRITCHAR.BACKUP.DATA') DAYS COUNT(5)
```

DFSMSrmm does not check quoted data set names for valid characters. Any string of up to 44 characters is accepted, except those that start with a blank or x'00'. Data set names without quotes must pass these data set naming rules:

### Creating a CDS record

When a CDS record is created, DFSMSrmm does not check quoted data set names or data set name masks for valid characters; any string of up to 44 characters is accepted except those which start with a blank or x'00'. Unquoted data set names and data set name masks must pass the data set naming rules which are described here one time and are as follows:

- ▶ A data set name can be one or more qualifiers.
- ▶ Each qualifier is 1 to 8 characters, the first of which must be alphabetic (A to Z) or national (# @ \$). The remaining seven characters are either alphabetic, numeric (0 - 9), national, or a hyphen (-).
- ▶ Qualifiers are separated by a period (.).
- ▶ DFSMSrmm adds your TSO PROFILE PREFIX value as the high-level qualifier.
- ▶ The data set name must not include a member name.

In addition data set name masks must pass the following data set mask naming rules:

- ▶ You can use \*, %, or ~ in a data set name mask.
- ▶ \*

  - A single \* represents a single qualifier of any number of characters.
  - A single \* when used within a qualifier represents zero or more characters.
  - More than one single \* can be used within a qualifier as long as a character precedes or follows the \*.

- ▶ .\*\*

  - represents zero or more qualifiers. At the end of the mask, \*\* indicates to ignore any remaining characters.

- ▶ \*\*

  - indicates to select all data sets. You can use this mask to define a vital record specification that sets your installation default retention criteria for data sets that are not covered by other vital record specifications.

- ▶ % (percent sign)

  - A place holder for a single character.

- ▶ ~ (not sign)

  - A place holder for a single character. The ~ has special meaning in a VRS data set name mask and is used to specify a pseudo-GDG data set name.

## 23.6.7 Data set names in RMM subcommands

When an existing record in the CDS is being processed, the checks are as follows:

- ▶ The name is 1 to 44 characters, enclosed in quotes if any special characters are included. If the data set name is not enclosed in quotes PROFILE PREFIX is applied but there is no check against data set naming or data set mask naming rules.
- ▶ Data set names are validated differently when:
  - Creating new data sets.
  - Using existing data sets.

This differentiation is shown in Figure 23-39 and explain points 1, 2 and 3 in the diagram.

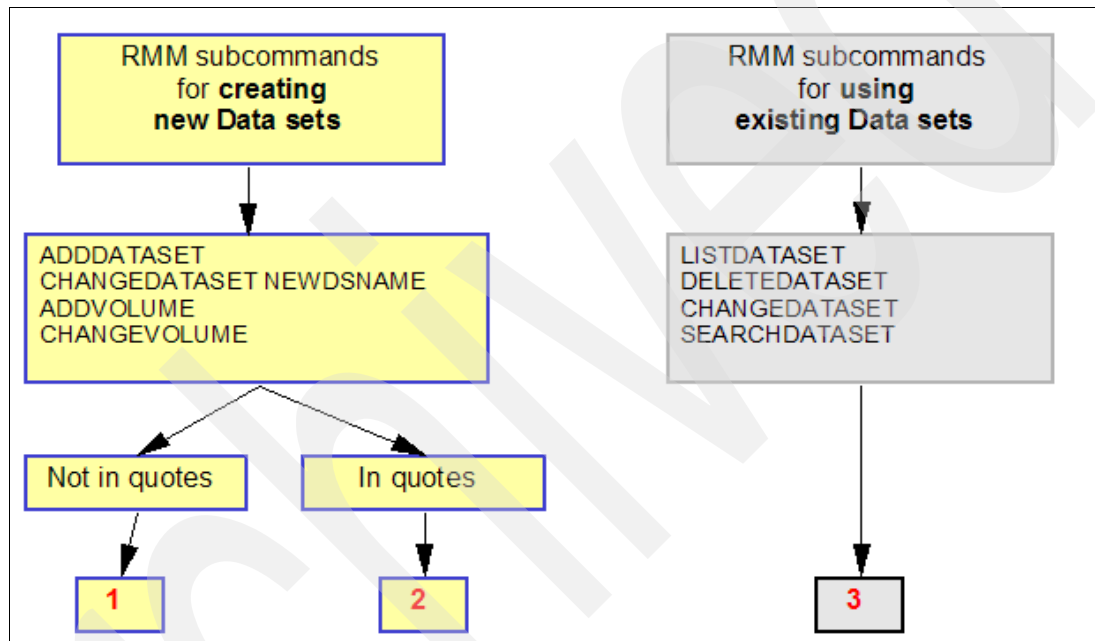


Figure 23-39 RMM subcommands

### Examples of RMM subcommands

With reference to Figure 23-39, following are some examples of specifying data set names:

1. Validation of an unquoted data set name when a data set is created.
  - a. The data set name must follow the standard MVS naming convention. The creation of these data set names will give the following results.

**AD TEST.NEW** Data set USERID.TEST.NEW will be added.

**AD spec/** IKJ56702I INVALID DATA SET NAME, spec/ial" will be issued.

2. Validation of a quoted data set name when a data set is created.
  - a. The data set name must not start with blank or null and the max length is 44 characters. The creation of these data set name will give the following results.

**AD '01/file'** Data set name '01/file' is added.

**AD ' text1'** IKJ56702I INVALID DATA SET NAME is issued because the data set name starts with a blank.



3. Validation of a data set name when an existing data set is used.
  - a. The maximum data set name length is 44 character. The display of the data set name will give the following results.

**LD '001/file'**                      The data set is listed.

**LD 'text1'**                              EDG3201I THE ENTRY IS NOT DEFINED TO DFSMSrmm is issued because the file was not created in point 2 above.

### Data set name masks

Data set names masks are validated differently when using data set name masks as follows:

- ▶ RMM subcommands for creating a VRS
- ▶ RMM subcommands for existing generic data set names and masks

This differentiation is shown in Figure 23-40 and explain points 1, 2, 3 and 4 in the diagram.

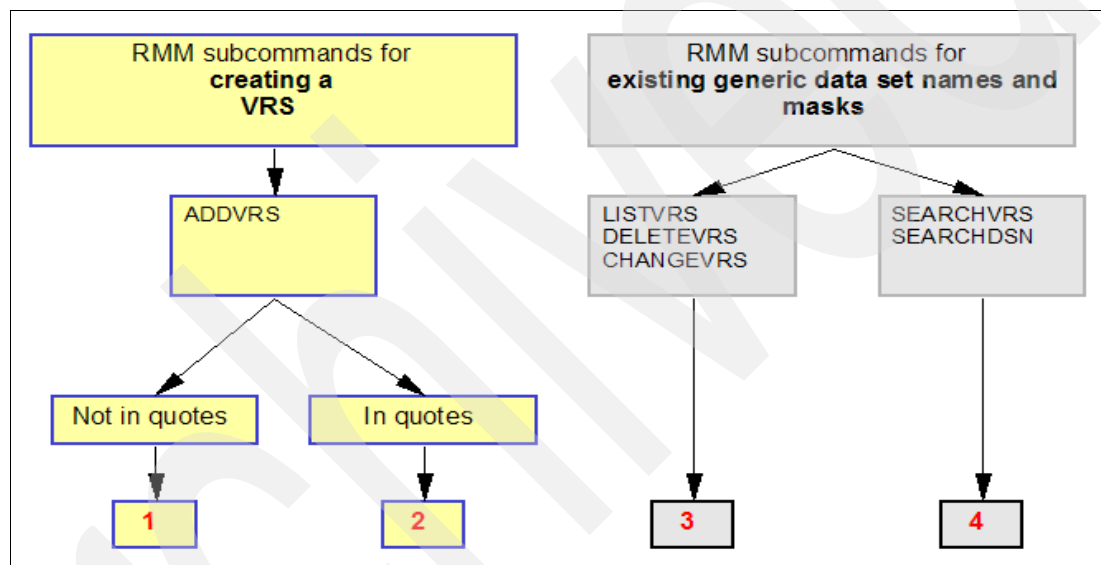


Figure 23-40 Masks in RMM subcommands

With reference to Figure 23-40, consider the following:

1. Validation of an unquoted data set name VRS at create time (AS).
  - a. Data set name mask must follow standard MVS naming convention.
  - a. Must follow common filtering and GDG rules (\*, \*\*, %, -).
2. Validation of a quoted Data Set Name VRS at create time (AS).
  - a. Must follow filtering rules for period and blank which are:
    - i. Data set name must not start with blank or null.
    - i. Must not start or end with period nor use consecutive periods.
  - a. Must follow common filtering and GDG rules (\*, \*\*, %, -).
  - a. Max length is 44 characters.
3. Validation of a Data Set Name Mask when an existing mask is used (LS, DS, CS).
  - a. Max length is 44 characters.

4. Validation of a Data Set Name Mask when an existing mask is used in SEARCH subcommands (SS, SD).
  - a. When a data set name mask does not follow common filtering or GDG rules or filtering rules for period and blank:
    - i. SEARCH subcommand is processed with fully qualified data set name.
  - a. Max length is 44 characters.

Figure 23-41 shows a few examples of data set name masks in RMM subcommands.

AS TEST.* ...	→ VRS USERID.TEST.* added
AS TE**.* ...	→ IKJ56702I INVALID DATA SET NAME MASK., TE**.*
AS spe/*. * ...	→ IKJ56702I INVALID DATA SET NAME MASK., spe/*. *
AS `spe/*. *` ...	→ VRS spec/*. * added
AS `TE**.*` ...	→ IKJ56702I INVALID DATA SET NAME MASK., `TE**.*`
AS `*.ab/*. *` ...	→ IKJ56702I INVALID DATA SET NAME MASK., `*.ab/*. *`
LS `spe/*. *` ...	→ VRS spec/*. * listed
DS `TE**.*` ...	→ EDG3201I THE ENTRY IS NOT DEFINED TO RMM
CS `*.ab/*. *` ...	→ EDG3201I THE ENTRY IS NOT DEFINED TO RMM
SS `spe/*. *` ...	→ VRS search output list
SD `TE**.*` ...	→ EDG3201I THE ENTRY IS NOT DEFINED TO RMM or full data set name TE**.* search output.

Figure 23-41 Examples of Masks in RMM

### Changes in vital record specification (VRSEL) processing

The new naming convention rules for data set names and data set name masks apply only on z/OS V1R9 and later releases. Caution should be used when creating new data set name VRSEs which include special characters or do not conform to the data set naming rules. The new masks may not be able to be processed by lower level systems, so ensure that VRSEL runs on a z/OS V1R9 or later release system.

In z/OS V1R9 DFSMSrmm VRSEL processing data set names are not translated to upper case. This is shown in Figure 23-42 on page 419.

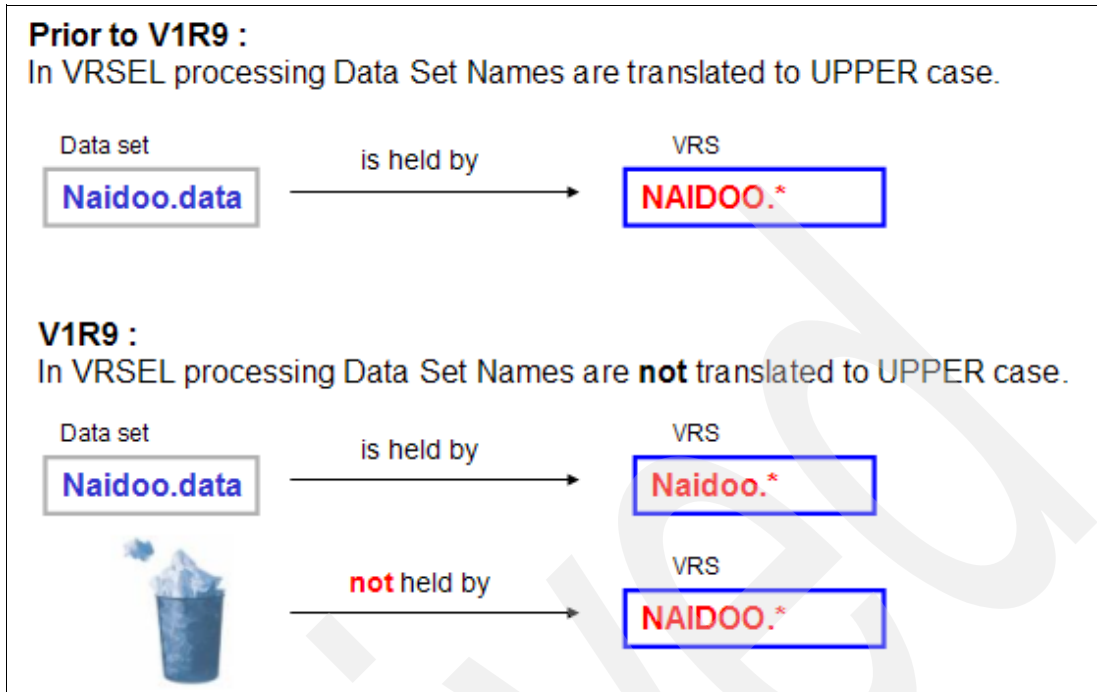


Figure 23-42 Changes in VRSEL processing

For data sets with lower or mixed case names that have been created prior to V1R9 the following migration aid must be honored. To check whether you have any lowercase or mixed case data set names that are on volumes that are VRS-retained, do the following:

- ▶ From a lower-level system with access to the z/OS V1R9 SYS1.SAMPLIB, copy the SAMPLIB member EDGGDSNM to your own report library (userid.REPORT.LIB). Although the report is ready to use, you can modify this EDGGDSNM sample to create your own tailored report.
- ▶ Using the DFSMSrmm report generator, generate the JCL to be used to create the report. If you do not have an existing report extract data set available to be used as input for the report, select the option to have the generated JCL create a new extract.
- ▶ Run the report. The report lists only those data sets that have lowercase or mixed case data set names that are on volumes that are VRS-retained. Note that when you run a report extract data set created for a release earlier than z/OS V1R9, there is no matching VRS information listed because earlier z/OS releases did not maintain this information for mixed case or lowercase data set names.

If the report indicates that you have lowercase or mixed case data set names on volumes that are VRS-retained, identify the VRSes that retain the data sets. If any of the matching DSNAMES VRSes contain uppercase letters that match the lowercase or mixed case letters in the data set names, the VRSes will no longer match the data set names in z/OS V1R9. You must create new VRSes or change the existing VRSes for the data sets. Data sets that match generic characters in the VRS data set name masks are not affected.

**Note:** While you run VRSEL vital record processing on releases below z/OS V1R9, you should not delete any VRSes that retain data sets with lowercase or mixed case letters, or you might lose data.

Figure 23-43 on page 420 is an example output of the migration aid (EDGGDSNM).

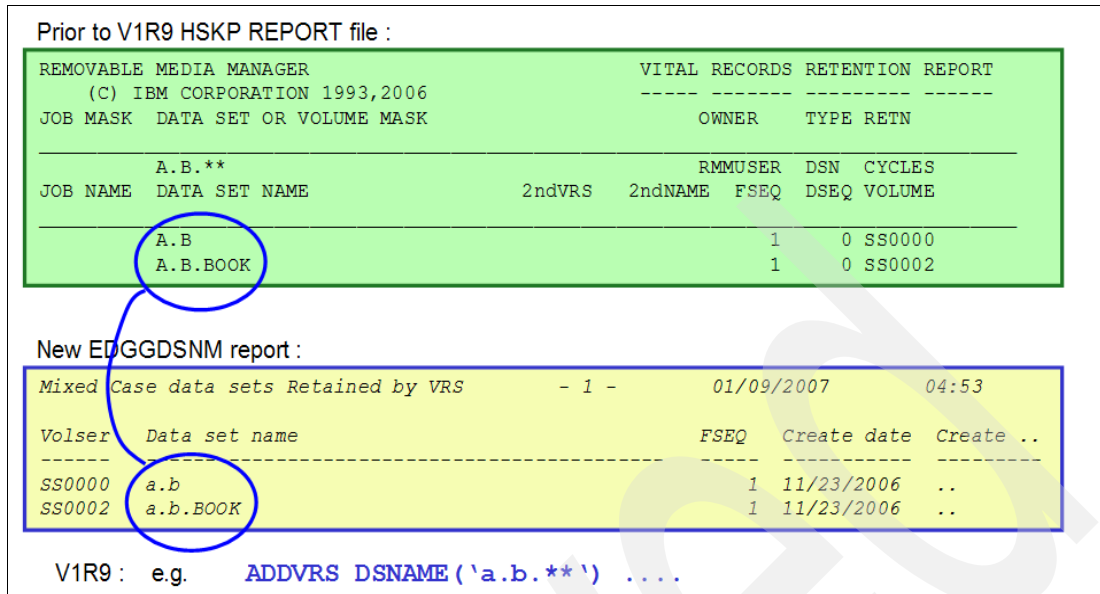


Figure 23-43 Example output of migration aid

### 23.6.8 Shared parmlib support

Some information in the EDGRMMxx parmlib may need to be specific to a subset of your systems. For example, the REJECT or VLPOOL entries may need to be different across systems. To enable this information to be handled on a system by system basis you can specify a second parmlib member to be used.

#### Visual overview

Use the MEMBER operand in the primary DFSMSrmm parmlib to identify a second parmlib member that contains overriding or additional parmlib options. Some information in the EDGRMMxx parmlib member may need to be specific to a subset of your systems. For example, the REJECT or VLPOOL entries need to be different as shown in Figure 23-44 on page 421.

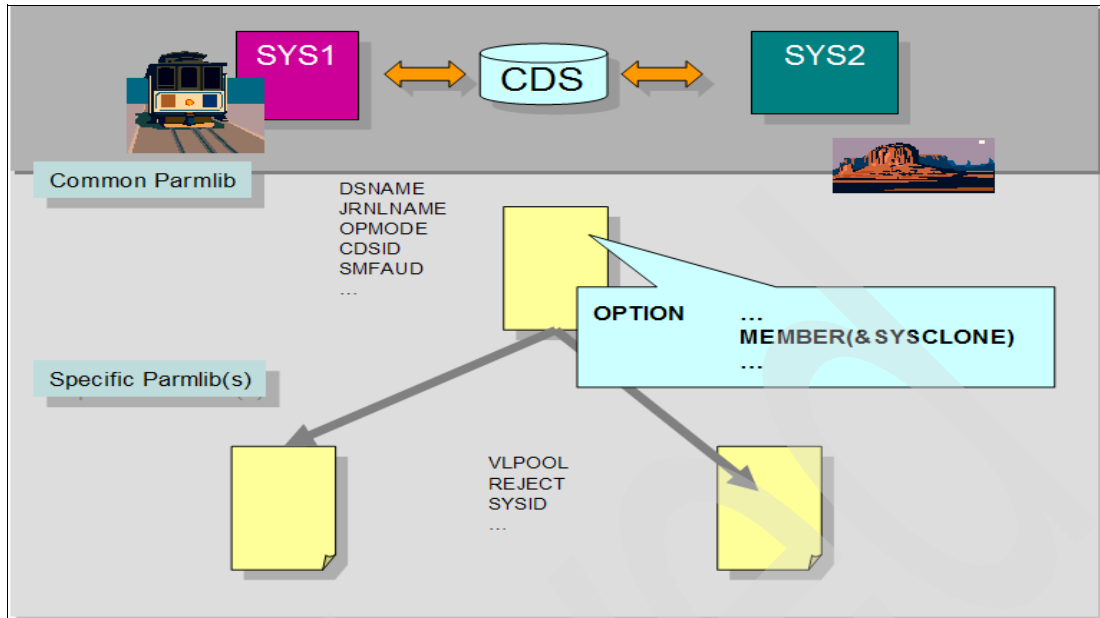


Figure 23-44 Second Parmlib overview

### OPTION MEMBER(xx)

The `parmlib_suffix` must be any 2 characters used as the suffix of the EDGRMMxx parmlib member name. Starting with z/OS V1R9, you can use system symbols (for example, `&SYSCLONE`) to enable easier sharing of the EDGRMMxx parmlib member. The system symbol must resolve to 2 characters used as the suffix of the EDGRMMxx parmlib member name.

DFSMSrmm processes the commands in the EDGRMMxx parmlib member and then if there is a second member named processes the second member. Any `OPTION` operands specified, other than `MEMBER` override the values set in the first member. Any other parmlib commands can add to, but not replace, update or duplicate any command from the first parmlib member. The processing by DFSMSrmm is as if all of the parmlib contents of both members had been specified in a single parmlib member, as follows:

```
OPTION DSNAME(DFRMM.CONTROL.DATASET)
        JRNLNAME(DFRMM.JOURNAL.DATASET)
        CDSID(SC70)
        MEMBER(70)
```

### LISTCONTROL OPTION

The `LISTCONTROL` `OPTION` output has been extended to display the second parmlib member name suffix as shown in Figure 23-45 on page 422.



- Any existing or user specified DCB record format parameters are now supported.
- ▶ Search command enhancements.
  - A new CONTINUE parameter was introduced for all search commands as a way to break down search results into manageable quantities.
  - There is a new STORAGEGROUP parameter for the Search Volume command.
- ▶ Report 17
  - A new report, REPORT 17, is added to the EDGRRPTE exec. It summarizes information for logical and stacked volumes to support stacked volume management.

### Changes in subcommand parsing

The product LEVEL specifies a software product's version. The form is VxxRxxMxx, indicating the Version, Release, and Modification level. Until now 'x' had to be numeric. With this release 'x' can be alphanumeric or national.

This change affects the following commands:

- ▶ AddProduct
- ▶ ChangeProduct
- ▶ ListProduct
- ▶ AddVolume
- ▶ ChangeVolume

**Note:** The default value is V01R01M00, Version 1, Release 1, Modification 0.

### NOSECLEVEL parameter

The SECLEVEL specifies a volume or data sets security class, that is defined for your installation. To remove the security classification for a volume or data set a new parameter was introduced with DFSMSrmm V1R9 called NOSECLEVEL. The command syntax is as shown in Figure 23-47.

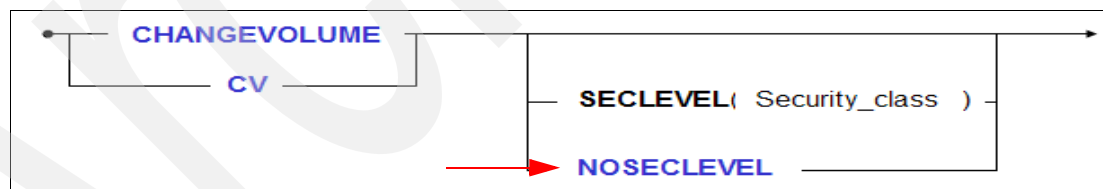


Figure 23-47 NOSECLEVEL parameter

**Attention:** Using ISPF panels you can reset the security level by clearing the SECLEVEL field. This will cause the appropriate execs to use the NOSECLEVEL parameter, when they generate the appropriate command.

The following commands are affected by this change:

- ▶ ChangeVolume
- ▶ ChangeDataset

## CLIST enhancements

With the CLIST parameter for all search commands you have the option to extend the results of your search to executable commands and to route these into a data set. Until now RMM generated a new CLIST data set (VB format, LRECL 255) for each command issued. With the new START/ADD parameters the user now has the choice to either add records to an existing data set or, as before, to replace the existing records in the CLIST data set.

The default value is START, what means write the records to the start of the data set, which will replace any existing records. If you specify ADD, the records will be appended at the end of the data set. In addition to the subcommand request, the disposition of the allocated RMMCLIST data set is taken into account. DISP=MOD overrides the START operand.

**Note:** If DISP=MOD is coded, records will always be appended to the RMMCLIST data set.

Like the CLIST parameter START / ADD can be specified in any SEARCH subcommand. The syntax of the START / ADD parameters is shown in Figure 23-48.

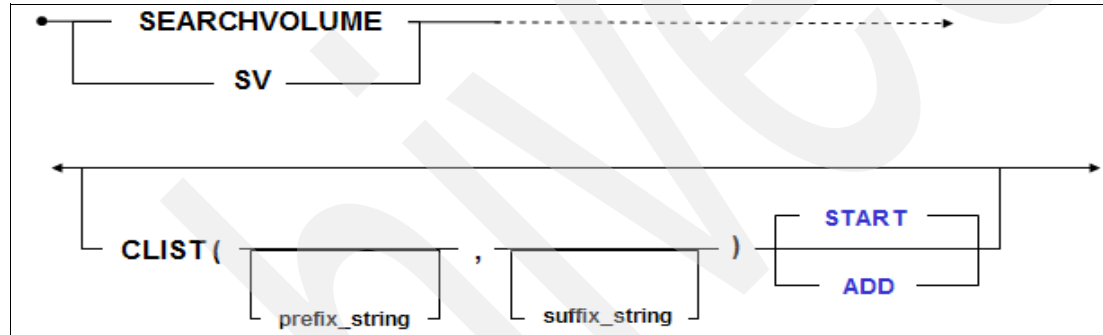


Figure 23-48 START/ADD parameter

## Sample JCL for SEARCH subcommand

Figure 23-49 on page 425 contains two JCL samples to issue a SEARCH subcommand and write data to a CLIST data set.

- ▶ In the first sample ddname RMMCLIST &SYSUID..CLIST.DATA is specified in the JCL and the RMM subcommand request specifies parameter ADD.
  - Data will be appended to data set &SYSUID..CLIST.DATA.
- ▶ In the second sample the same RMMCLIST data set is specified in the JCL and the subcommand request specifies parameter START. Since DISP=MOD is specified for the RMMCLIST data set, the result will be the same as in the first sample.
  - Data will be appended to data set &SYSUID..CLIST.DATA.



```

/*-----
/*  WILL APPEND RECORDS TO AN EXISTING DATA SET
/*-----
//CL1 EXEC PGM=IKJEFT01
//RMMCLIST DD DSN=&SYSUID..CLIST.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
    RMM SV OWNER(*) CLIST ADD

/*-----
/*  WILL APPEND RECORDS TO AN EXISTING DATA SET
/*-----
//CL1 EXEC PGM=IKJEFT01
//RMMCLIST DD DSN=&SYSUID..CLIST.DATA,DISP=MOD
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
    RMM SV OWNER(*) CLIST START

```

Figure 23-49 JCL samples

### User defined format of CLIST data set

Until now the format of the RMM CLIST data set was fixed to VB 255. With release z/OS V1R9 DFSMSrmm starts to honor the DCB attributes of an existing CLIST data set. This means that you can pre-allocate data sets in a format you prefer and RMM will leave this format unchanged.

The following restrictions apply:

- ▶ Pre-allocated data sets can be fixed or variable format
- ▶ LRECL must be at least long enough to contain the CLIST information for the record type you are searching.
- ▶ If LRECL is too short, RMM increases LRECL to the minimum needed. If this is impossible, message **EDG3360E** is issued.
- ▶ The maximum LRECL supported is 32760.
- ▶ If data set doesn't exist, defaults are LRECL 255, VB format.

### Search command enhancements

When you issue a subcommand that you know you may wish to continue, you specify the **CONTINUE** operand without any value. You use the **LIMIT** operand to specify how many search result entries you can manage each time you continue the search. After the first command you process the data returned and the continue\_information returned, and, if more records exist, you repeat the command (now with continue\_information) until all results are returned. This is shown in Figure 23-50 on page 426.

The continue\_information must be passed back to DFSMSrmm unchanged in order to continue the previous search. You should also specify the exact same subcommand unchanged, just changing the CONTINUE value on each additional command required.

## CONTINUE parameter

→ a new way to break down the results of a search into manageable quantities !

sample, to search through all volumes :

```
RMM SV VOLUME(*) OWNER(*) LIMIT(1000) CONTINUE
```

process the volumes and the continue\_information returned,

then repeat

```
RMM SV VOLUME(*) OWNER(*) LIMIT(1000) CONTINUE(continue_information)
```

until all results are returned

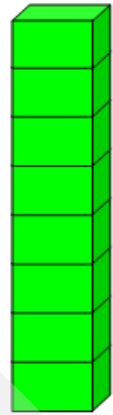


Figure 23-50 CONTINUE command

The content of the continue\_information depends on the type of search done. In Figure 23-51 you see a complete list of RMM search commands and the appropriate continue\_information. In general, the continue\_information contains a valid key for the specific RMM resource searched for. For example:

- ▶ If you are searching for volumes, the continue\_information will contain a volser.
  - VOLUME('V10035') or VOLUME('V10071').
- ▶ if you are searching for data sets, the continue\_information will contain data set name plus volser plus file sequence number.
  - DSNAME('RMMUSER.DATA36')VOLUME('V10003')FILESEQ(6).

### continue\_information - content depends on the type of your search

<b>SearchBin</b>	– BIN(bin_number)	– MEDIANAME(media)	– LOCATION(loc_name)
		– STORE(builtin_store)	
<b>SearchDataset</b>	– DSNAME(dsname)	– VOLUME(volser)	– FILESEQ(seq)
<b>SearchOwner</b>	– OWNER(ownerid)		
<b>SearchProduct</b>	– NUMBER(product_number)	– LEVEL(version_number)	
<b>SearchRack</b>	– RACK(rack_number)	– MEDIANAME(media)	
<b>SearchVolume</b>	– VOLUME(volser)		
<b>SearchVRS</b>	– VOLUME(volser)		
	– NAME(vrs_name)		
	– DSNAME(datasetmask)		– JOBNAME(jobname mask)

Figure 23-51 CONTINUE\_INFORMATION information

If in Figure 23-52 we see a typical SEARCH example of using the CONTINUE parameter. If you issue a search command from TSO environment, you get a table of resources returned and at the end of this table, there will be messages:

- ▶ EDG3203I SEARCH COMPLETE - MORE ENTRIES MAY EXIST
- ▶ EDG3012I xxxx ENTRIES LISTED

If you specify the CONTINUE parameter, there will be a new additional message now:

- ▶ EDG3025I continue\_information

You can use this information to build a new search command, pulling in the next RMM resources in sequence.

```
==> RMM SV OWNER(*) LIMIT(1000) CONTINUE
```

Volume	Owner	Rack	Assigned date	Expiration date	Location	Dsets	St Act	Dest.
S11003	BERND		2006/328	2006/333	SHELF	0	M	
S11008	BERND		2006/328	2006/333	SHELF	0	M	
S11009	BERND		2006/328	2006/333	SHELF	0	M	

```
EDG3203I SEARCH COMPLETE - MORE ENTRIES MAY EXIST
EDG3012I 1000 ENTRIES LISTED
EDG3025I VOLUME('S11009')
***
```

```
==> RMM SV OWNER(*) LIMIT(1000) CONTINUE(VOLUME('S11009'))
```

Figure 23-52 CONTINUE\_INFORMATION volume

When you issue a search command from REXX and you request results as REXX variables, the EDG@CONT variable contains the character string you should use to continue the subcommand. When you use the API, and data is requested back as SFI the CONT SFI contains the character string you should use. This is shown in Figure 23-53.

REXX environment :		variable EDG@CONT			
Variable Name	Data Type	Variable Length	Variable Values	Commands	
CONT	character (variable length)	84	Search Continue information	All Searchxxx commands	

API interface :		SFI CONT			
SFI Name	SFI number	SFI Data Type	SFI Length	SFI Data RC and RS	Commands
CONT	X'057000'	7 character (variable length)	92	Search Continue information RC=4 RS=2	All Searchxxx commands

Figure 23-53 REXX and API search

When you use the High Level Language API or the RMM Web Service and data is requested back as XML, you will find message EDG3025I, containing the continue\_information, in the group at the end of the returned resources as follows:

```
</VOLUME>
<INFO>
<RTNC>4</RTNC>
<RSNC>2</RSNC>
<MSGT>EDG3012I 1000 ENTRIES LISTED </MSGT>
<MSGT>EDG3025I VOLUME('A11021') </MSGT>
</INFO>
```

Any continue\_information you enter is handled as is by the RMM command processor. The case you specify is retained and used by RMM, so that CONTINUE(VOLUME('ab1000')) and CONTINUE(VOLUME('AB1000')) are different continuation points as shown in Figure 23-54. RMM does not change any values in the CONTINUE operand to upper case.

The last command in our example requests 1000 resources, but only 27 are returned. This means there are no more resources to return. In this case there is NO continue\_information returned. No EDG3025I is issued and EDG@CONT will also be blank.

```
SV VOLUME(*) OWNER(*) LIMIT(1000) CONTINUE CLIST NOLIST START
EDG3012I 1000 ENTRIES LISTED
EDG3025I VOLUME('ab1000')

SV VOLUME(*) OWNER(*) LIMIT(1000) CONTINUE(VOLUME('ab1000')) CLIST NOLIST ADD
EDG3012I 1000 ENTRIES LISTED
EDG3025I VOLUME('AB1000')

SV VOLUME(*) OWNER(*) LIMIT(1000) CONTINUE(VOLUME('AB1000')) CLIST NOLIST ADD
EDG3012I 27 ENTRIES LISTED
```

Figure 23-54 Case sensitive CONTINUE

**Note:** RMM returns continue\_information values delimited with quotes.

You will see no change in the ISPF panels for use of the CONTINUE parameter but under the covers the search execs will exploit this new feature. When the LIMIT value is either set to \* or is larger than 2000, the execs will use CONTINUE as shown in Figure 23-55 on page 429.

The EDGRxSCH execs will internally handle the panel request for a limit >2000 and specify LIMIT on the SEARCHxxxx subcommand accordingly, using LIMIT(2000) until the last SEARCH when LIMIT is set to panel\_specified\_limit-(n\*2000).

```

                                DFSMSrmm Volume Search

Volume . . . . . *           May be generic. Leave blank for all volumes
Owner . . . . . *           Owned by a specific user. Default is your userid
Job name . . . .           May be generic
Limit . . . . . 7000        Limit search to first nnnn volumes

➔ SV VOLUME(*) OWNER(*) LIMIT(2000) CONTINUE
  SV VOLUME(*) OWNER(*) LIMIT(2000) CONTINUE(continue_information_1)
  SV VOLUME(*) OWNER(*) LIMIT(2000) CONTINUE(continue_information_2)
  SV VOLUME(*) OWNER(*) LIMIT(1000) CONTINUE(continue_information_3)

```

Figure 23-55 LIMIT larger than 2000

### STORAGEGROUP parameter

For the SearchVolume command the new STORAGEGROUP parameter was introduced.

- ▶ RMM SEARCHVOLUME STORAGEGROUP(storage\_group\_name)
  - You can specify the storage group name to select a subset of volumes based on the assigned storage group name.
  - A storage group name is one-to-eight alphanumeric characters.
  - A storage group name can be a value that matches to a VLPOOL NAME value but does not need to be defined on a VLPOOL definition.
  - DFSMSrmm accepts the abbreviation STORGRP.
  - Use the STORAGEGROUP operand to build lists of exportable volumes which are in the same VTS physical volume pool.

### REPORT17

When a stacked volume, containing exported logical volumes, is ejected from the library, as the logical volumes expire, RMM places the volumes in a "pending release" state and then when the logical volumes are imported into the library, RMM completes the return to scratch process enabling the volumes to be reused. As the exported logical volumes expire, you need the ability to do off-site stacked volume management so you can determine when to bring a stacked volume back on-site for possible reuse. RMM has enough information in its database for you to create and run reports, however, a specific stacked volume management report did not exist.

With z/OS V1R9, RMM provides a new stacked volume management report for customers to customize and run, that includes the ability to report on the percentage of active data on a stacked volume and to also report on the percentage of active logical volumes on a stacked volume. A new report, REPORT17, is added to EDGRRPTE reporting exec which summarizes information for logical and stacked volumes. The stacked volumes are presented in order of increasing percentage of active number of volumes and percentage used. The least used stacked volumes are listed first. Figure 23-56 on page 430 shows an example of a REPORT17 report.

DFSMSrmm INTERNAL USE ONLY										Inventory of Stacked Volumes by Percent Active		PAGE -	1
EDGRPT17												DATE -	2005/346
												TIME -	06:17:18
Volume	%	#	#	%	Media	Retention	V Location	Store	Export	Export	Home		
Serial	Act	Active	Logical	Use Capacity	Type	Date	R Name	Date	Date	Time	Location		
A00277	5	1	20	10	20000	EHFCT	2006/011	N	SHELF	2005/346	061712	SHELF	
A00257	10	1	10	5	286102	EFMT1	2006/011	N	SHELF	2005/346	061711	SHELF	
A00237	33	1	3	50	286102	EFMT1	2006/054	N	SHELF	2005/346	061710	SHELF	
A00217	50	2	4	80	20000	EHFCT	2006/001	N	SHELF	2005/346	061709	SHELF	
A00197	67	2	3	50	286102	EFMT1	2005/349	N	SHELF	2005/346	061708	SHELF	
A00177	67	2	3	67	286102	EFMT1	2005/349	N	SHELF	2005/346	061707	SHELF	
A00157	100	3	3	100	286102	EFMT1	2006/001	N	SHELF	2005/346	061706	SHELF	

Figure 23-56 REPORT17 printout

The report columns are explained as follows:

<b>% Act</b>	Percentage of the contained logical volumes which are active.
<b># Active</b>	The number of active logical volumes. Active logical volumes are all those which are neither scratch nor pending release.
<b># Logical</b>	The number of contained logical volumes.
<b>% Use</b>	The approximate percentage of active data.
<b>Capacity</b>	The size of the stacked volume in MB.
<b>Retention Date</b>	When VRS retained this is the VRS calculated retention date otherwise it is the latest expiration date of all contained active volumes.
<b>Export Date</b>	The date when the stacked volume was exported from a VTS.
<b>Export Time</b>	The time when the stacked volumes was exported from a VTS.

### 23.6.10 3592 Model E05 software support

Using the existing media types (MEDIA5, MEDIA6, MEDIA7 and MEDIA8) and the two extended length future media types (MEDIA9 and MEDIA10), an encryption enabled 3592 Model E05, reads and writes using the new Enterprise Encrypted Format 2 (EEFMT2) recording technology. It can also read and write using the Enterprise Format 1 (EFMT1) and Enterprise Format 2 (EFMT2) non-encrypted recording technologies.

In order to request EEFMT2 in the stand-alone and in the system-managed IBM tape library environment, a DFSMS dataclass must be used which specifies EEFMT2 as its recording technology, otherwise EFMT2 is the default recording technology that is used. Dataclass can also be used to request the EFMT1 recording format and to explicitly request the EFMT2 format. A mix of recording formats is not supported on the same tape cartridge. An enhanced 3592 Model E05 that does not have the encryption feature enabled, can only read and write in the non-encryption formats (EFMT1 and EFMT2) which is the same as the base 3592 Model E05. In a mixed 3592 environment, new microcode is also needed for the 3592 Model J and the base 3592 Model E05 so that it recognizes a volume with the new EEFMT2 recording technology.

#### Change summary

New recording format external for encrypted media (EEFMT2).

- ▶ enterprise encrypted format 2.

Encryption requested through DFSMS data class.

- ▶ through specification of the new recording format EEFMT2 (EE2).
- ▶ if the encrypted format (EEFMT2) is not specified, the non-encrypted formats EFMT1 or EFMT2 are used.

Encryption is supported with all existing 3592 media types.

- ▶ MEDIA5 – MEDIA10 (including WORM media).

Existing data class options, performance scaling and performance segmentation is also supported with encryption.

Figure 23-57 shows where you specify EEFMT2 in the DFSMS data class.

```
Panel Utilities Scroll Help
-----
                                DATA CLASS ALTER                                Page 3 of 5
Command ===> _____
SCDS Name . . . . : SYS1.SMS.SCDS
Data Class Name : DC3592

To ALTER Data Class, Specify:

Media Interchange
Media Type . . . . . _____ (1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or blank)
Recording Technology . . . . . _____ (18,36,128,256,384,E1,E2,EE2 or blank)
Performance Scaling . . . . . _____ (Y, N or blank)
Performance Segmentation . . . . . _____ (Y, N or blank)
Block Size Limit . . . . . _____ (32760 to 2GB or blank)
Recorg . . . . . _____ (KS, ES, RR, LS or blank)
Keylen . . . . . _____ (0 to 255 or blank)
Keyoff . . . . . _____ (0 to 32760 or blank)
CIsze Data . . . . . _____ (1 to 32768 or blank)
% Freespace CI . . . . . _____ (0 to 100 or blank)
CA . . . . . _____ (0 to 100 or blank)
Use ENTER to Perform Verification; Use UP/DOWN Command to View other Panels;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.
```

Figure 23-57 EEFMT2 in recording technology field

### 23.6.11 Migration and coexistence considerations

The following APARS must be researched and installed on the relevant systems.

**RMM preconditioning** APAR OA16523 (V1R6 – V1R8)

**RMM toleration** APAR OA16524 (V1R4 – V1R8)

**RMM tape encryption** APAR OA15698 (V1R6 – V1R7)  
APAR OA17574 (V1R8)

- ▶ The toleration APAR OA16524 is based on OA15623 (preconditioning).
- ▶ The tape encryption APARs OA15698 and OA17574 are based on OA16524 (toleration).
- ▶ For the installation on a client/server RMMplex is mandatory:
  - Install pre-conditioning PTF on all systems of the RMMplex.
  - Then install the toleration PTF on all systems.
  - ONLY then the RMM tape encryption PTF can be installed.
- ▶ Pre-Conditioning and Toleration APARs contain a ++HOLD(MULTSYS) text, which describes this dependency.

### Label anomaly processing

APAR OA18455 was created to prevent DFSMSrmm from setting the PENDING RELEASE ACTION INIT flag, when a “servo track format error” is detected. The tape is not blocked anymore by DFSMSrmm, with the message:

- ▶ EDG4033I VOLUME xxxxxx REJECTED. THE VOLUME IS WAITING TO BE REINITIALIZED.

## 23.7 Network File Systems (NFS) enhancements

Network File System (NFS) is a base element of z/OS, that allows remote access to z/OS host processor data from workstations, personal computers, or any other system on a TCP/IP network that is using client software for the Network File System protocol.

The following topics will be discussed:

- ▶ 24-bit Addressing Relief
- ▶ Multi TCP/IP Stack Support
- ▶ AddDS Operator Command
- ▶ RACF Data Labeling
- ▶ NFS v4 Client Support
- ▶ Client Attribute syntax
- ▶ Server CTrace Upgrade
- ▶ Terminal ID based restricted MVSLOGIN

### 23.7.1 24-bit addressing relief

Currently all z/OS NFS Server tasks that interact with z/OS MVS data sets have their stacks and heaps defined below the 16MB line. This puts major constraints on the number of z/OS NFS Server tasks that can exist.

In z/OS V1R9 NFS task structures have been modified to permit stacks and heaps above the 16 MB line. This will enable NFS server tasks to allow more parallel NFS request processing.

**Note:** There will be a performance improvement due to more parallel NFS request processing.

### 23.7.2 Multi TCP/IP stack support

Currently there is a restriction limiting the z/OS NFS RPCSEC security mechanism to only support a single IP address, limiting it to a single TCPIP stack. z/OS NFS Server V1R9 will now be able to successfully interact with multiple TCPIP Stacks, including VIPA (Virtual IP Assignment).

z/OS supports the ability to have multiple TCPIP stacks on a single system, including Virtual IP Assignment (VIPA). These multiple stacks and VIPA are useful for providing enhanced system reliability, providing alternate message paths should a given stack break.

The z/OS NFS RPCSEC security mechanism has been modified to allow it to successfully interact with multiple TCPIP Stacks.



### 23.7.3 Usage and invocation

To set up the multiple TCP/IP stack support you must do the following:

- ▶ The BPXPRMxx parmlib member must be updated to include all stacks.
  - One Stack must be marked as DEFAULT.
- ▶ Portmapper/rpcbind must be defined as generic server.
  - No stack affinity is allowed.
- ▶ NFS server must be defined as a generic server.
  - No stack affinity is allowed.
- ▶ Must have a TCPIP profile defined for each stack.
- ▶ For Kerberos, since each stack has its own hostname and IP a keytab must be created for each stack.

**Note:** For more information about configuring multiple TCP/IP stacks, see z/OS UNIX System Services Planning and z/OS Communications Server: IP Configuration Reference.

The CINET configuration in the BPXPRMxx parmlib member to start a single NFS server in a multi stack environment is shown, as follows:

```
FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXCINT)
  NETWORK TYPE(CINET)
    DOMAINNAME(AF_INET)
    DOMAINNUMBER(2)
    MAXSOCKETS(64000)
    INADDRANYPORT(4901)
    INADDRANYCOUNT(100)
  NETWORK TYPE(CINET)
    DOMAINNAME(AF_INET6) /* activate IPV6 */
    DOMAINNUMBER(19)
SUBFILESYSTYPE TYPE(CINET) NAME(TCPIPRX) ENTRYPOINT(EZBPFINI) DEFAULT
SUBFILESYSTYPE TYPE(CINET) NAME(TCPIPRY) ENTRYPOINT(EZBPFINI)
```

**Note:** Common Inet Sockets is intended to be used only if multiple network socket file systems (such as 2 TCPIP's) are to be active at one time. There is a performance degradation with using Common Inet Sockets with just a single sockets physical file system.

### 23.7.4 AddDS operator command

The new **AddDS** operator command will allow a z/OS System Programmer to specify a replacement for one of the NFS control data sets. This is useful in the event one of these control data sets becomes unusable.

- ▶ Using the **AddDS** operator command, a z/OS system programmer can replace the following types of NFS control data sets:
  - Lock data sets
  - Mount handle database data sets

- ▶ The **addds** command requires the existing NFS control data set to be freed before this command is issued.
  - This can be achieved using the new **freeds** operator command.
- ▶ It is only possible to free the currently inactive data set of the MHDB, or LDB, pair. Therefore, if the active data set is the one to be freed, it is necessary to first swap the data set pair.
  - This can be accomplished with a new **swapmhdb**, or **swapldb**, command. This command swaps the active and inactive data sets in the database. Once this is done, it is then possible to free the previously active data set.

### Command syntax

The syntax of the **addds** command is:

```
MODIFY mvsnfs,ADDDS=ddname(dsname)
```

Where:

- ▶ **mvsnfs** is the name of the procedure in the system PROCLIB that was used to start the server.
- ▶ **ddname** is the ddname of the NFS server control data set which is to be replaced.
  - The valid ddnames are FHDBASE, FHDBASE2, LDBASE and LDBASE2.
- ▶ **dsname** is the name of the lock data set or mount handle data set to be enabled for use by the z/OS NFS server.

The syntax of the **freeds** command is:

```
MODIFY mvsnfs,FREEDS=ddname
```

Where:

- ▶ **mvsnfs** is the name of the procedure in the system PROCLIB that was used to start the server.
- ▶ **ddname** is the ddname of the NFS server control data set which is to be freed.
  - The valid ddnames are FHDBASE, FHDBASE2, LDBASE and LDBASE2.

The syntax of the swap commands is:

```
MODIFY mvsnfs,SWAPMHDB
```

Where:

- ▶ **mvsnfs** is the name of the procedure in the system PROCLIB that was used to start the server.

## 23.7.5 RACF data labeling

The z/OS NFS Server currently does not provide support for the RACF Data Labeling option **MLNAMES** (also known as Name-Hiding). z/OS V1R9 NFS Server will now provide this support, enabling active MVS data set names to be hidden from NFS users who do not have at least **READ** access to the data sets.

### MLNAMES overview

- ▶ When this option is active MVS data set names will be hidden from NFS users who do not have at least **READ** access to the data sets. Therefore, it may change the contents of an MVS data set index list when requested via the "ls -l" command.

- ▶ The z/OS NFS server only supports this option in SAF or SAFEXP SECURITY mode.
- ▶ This function only applies to MVS data set access, not to z/OS Unix file access.
- ▶ The name-hiding function can degrade system performance because it requires authorization checks for every object for which a non-SPECIAL user attempts to list the name.

This option is activated by RACF command:

```
SETROPTS MLNAMES
```

and is deactivated by RACF command:

```
SETROPTS NOMLNAMES
```

### 23.7.6 NFS v4 client support

Unlike previous z/OS releases, z/OS NFS Client V1R9 now provides support for the NFS v4 Protocol. It exploits the enhancements in the v4 protocol in the areas of internet performance and cross-platform inter operability

#### NFS V4 protocol

The Network File System (NFS) version 4 is a distributed file system protocol which owes its heritage to NFS protocol version 2 and 3. Unlike earlier versions the NFS V4 protocol supports traditional file access while integrating support for file locking and the mount protocol. In addition, support for strong security, compound operations, client caching and internationalization have been added. Attention has been applied to make NFS V4 operate well in an internet environment.

The client application interface to the NFS Client does not change with the NFS version 4 protocol. It is the NFS Client's responsibility to convert the application requests to the appropriate V4 requests. From an NFS Client perspective, the functional characteristics of this protocol are partitioned into several major areas.

- ▶ Security
- ▶ Name Space and Pseudo File Systems
- ▶ Client ID
- ▶ Locking
- ▶ Attribute

#### Restrictions

The z/OS NFS v4 Client in V1R9 does **not** currently include:

- ▶ RPCSEC\_GSS Security
- ▶ Locking

The NFS Version 4 Protocol is an industry wide standard. Therefore, the z/OS NFS V1R9 NFS Client, which is implementing that standard, should successfully communicate with any NFS Server (e.g. AIX/SUN) which adheres to that standard.

- ▶ Since this standard requires continued support of the NFS Version 2 and Version 3 Protocols as well, the z/OS NFS V1R9 Client should also successfully communicate with any NFS V2/V3 Server.

## Mount command

The NFS V4 protocol has been added to the `vers` parameter of the `mount` command.

- ▶ `mount -o vers=n`

Where “n” specifies the NFS protocol version to be used

- ▶ 2 or 3 for Linux.
- ▶ 2, 3 or 4 for others.

### 23.7.7 Client Attribute syntax

A new `public` option and `stringprep` option has been added to the z/OS NFS client attribute syntax.

**public** Forces the use of the public file handle when connecting to the NFS server. This option is valid only during mount processing.

**Note:** The `public` keyword is valid only for the NFS version 4 protocol.

**stringprep(Y/N)** Specifies whether z/OS NFS Client is to enable or disable `stringprep` normalization. `Stringprep` normalization is the NFS version 4 internationalization function for converting inbound strings to UTF-8 format.

**Note:** The `stringprep` attribute default value is “N”.

### 23.7.8 Server Ctrace upgrade

The NFS Server Ctrace function has been upgraded to use the common underlying support functions with the Client. This upgrade enables the NFS Server to exploit all of the function enhancements which were implemented for the NFS Client and improves serviceability.

#### Enhancements with z/OS V1R9

The following enhancements have been made:

- ▶ The “`MODIFY mvsnfs,FLUSHCTR`” operator command has been removed. The function performed by this command will now be performed automatically when the Component Trace external writer is stopped. It will no longer be necessary to explicitly execute this command first to flush the buffer.
- ▶ NFS component trace status and active options can be **displayed** by using the MVS `display` command:
  - `DISPLAY TRACE, COMP= mvsnfs` (for the NFS server)
  - `DISPLAY TRACE, COMP= mvsnfsc` (for the NFS client)

Figure 23-58 on page 437 shows the result of a `DISPLAY TRACE` command for the NFS server.

```

DISPLAY TRACE,COMP=NFSMVS
IEE843I 13.24.41 TRACE DISPLAY 240
      SYSTEM STATUS INFORMATION
ST=(ON,0256K,01536K) AS=ON BR=OFF EX=ON MO=OFF MT=(ON,064K)
COMPONENT      MODE BUFFER HEAD SUBS
-----
NFSMVS         ON   0010M
ASIDS          *NOT SUPPORTED*
JOBNAMES       *NOT SUPPORTED*
OPTIONS        FFDC          INFO          WARNING      ERROR
WRITER         *NONE*

```

Figure 23-58 DISPLAY TRACE command

- ▶ Four new trace record types have been added:
  - Buffer**                    Buffer Management (i.e BUFNODE)
  - Lock\_Request**            Control Block Lock Requests
  - Lock\_Result**             Control Block Lock Request results (lock granted, lock in use, error, etc.)
  - Lock\_Release**            Control Block Lock Release
- ▶ Trace command option values were added for the new record types listed above.
  - Msg**                        Error, Warning and Info Records
  - Wait**                      Suspend and Resume record types
  - Queue**                     Schedule and Dispatch record types
  - Lock**                        Lock\_Req, Lock\_Result and Lock\_Release record types
- ▶ The ability to turn off individual options was added.
  - An option can be turned off by preceding the option value with a minus sign (e.g. OPTIONS=(-GENERAL)).
  - Options are processed from left to right, first processing all values to turn on options and then processing all values to turn off options. Thus all options except network can be turned on with the following options specification: OPTIONS=(ALL,-NETWORK).
  - If an options value of “-ALL” is specified, the option revert back to a setting of MIN.

**Note:** A value of “-MIN” is invalid and will be ignored.

- ▶ The trace buffer size can be specified either by the BUFSIZE value in the startup parmlib member, or it can continue to be specified via the “DSPS=” startup parameter. The new default BUFSIZE value of 10 MB also applies to the NFS Client.

**Note:** The size of the trace buffer may still not be altered after startup.

- ▶ NFS component trace buffers captured in an MVS dump, or a Component Trace data set, may be viewed using the IPCS CTRACE command.

### Migration and coexistence considerations

APAR OA18325 must be applied to z/OS V1R7 and R8. The APAR is for rollback of the NFS V1R9 support to allow server ctrace to start even if the BUFSIZE parameter is present. The

BUFFER parameter is added to the CTRACE DEFINE macro and the BUFSIZE parameter in the CTINFSxx parmlib member is then ignored.

### 23.7.9 Terminal ID based restricted MVSLOGIN

When the z/OS NFS Server is used in SECURITY (SAF or SAFEXP) mode, it is necessary for users on NFS clients to issue an NFS Client Enabling Utility “**MVSLOGIN**” command from the NFS client system before they can access any files on the NFS Server. Normally, assuming the user has a valid z/OS userid and password, this is not a problem and will successfully provide the user with access to the z/OS system through NFS.

However, with the appropriate RACF configuration specifications, the z/OS NFS server also provides the ability to restrict MVSLOGINS based on an NFS client's IP address. In order to support this capability, the z/OS NFS server transforms an NFS client's IP address into an 8-byte character string, which is then used as the Terminal ID (termid) for that NFS Client. Each decimal number of the IP address is transformed into two hex digits. For example the following IP addresses are transformed into hexadecimal numbers:

12.15.16.32	0C0F1020
9.157.161.12	099DA10C

To use this capability, the z/OS system administrator must do the following:

- ▶ Activate the RACF class “TERMINAL” with this RACF command.

```
SETROPTS CLASSACT(TERMINAL)
```

- ▶ Define the proper resource in the TERMINAL class with this RACF command.

```
RDEFINE TERMINAL termid UACC(NONE)
```

**Note:** If a termid value of **099DA10C** is specified, then non-SPECIAL users on the NFS client with IP address **9.157.161.12** will not be able to execute the MVSLOGIN NFS Client Enabling Utility.

- ▶ Grant permission to some users (for example, CIANKA and KEEGAN) from the NFS client with IP address 9.157.161.12 to successfully execute the MVSLOGIN NFS client enabling utility with this RACF command.

```
PERMIT 099DA10C CLASS(TERMINAL) ID(CIANKA KEEGAN) ACCESS(ALTER)
```

## Large format data sets

Prior to z/OS V1R7, most sequential data sets were limited to 65535 tracks on each volume, although most hardware storage devices supported far more tracks per volume. To support this hardware capability, z/OS V1R7 and above allows you to create new large format data sets, which are physical sequential data sets with the ability to grow beyond the previous size limit. Large format data sets reduce the need to use multiple volumes for single data sets, especially very large ones like spool data sets, dumps, logs, and traces. Large format data sets can be either cataloged or uncataloged and SMS-managed or not.

In this chapter, the following topics are discussed:

- ▶ Large track data set overview
- ▶ TSO transmit and receive large format data sets > 64K tracks
- ▶ TSO **PRINTDS** command
- ▶ REXX and CLIST LISTDSI function
- ▶ Enhanced I/O capability in TSO/E for CLIST and REXX
- ▶ Messages
- ▶ Migration and coexistence considerations

## 24.1 Large format data set overview

As mentioned, large format data sets are physical sequential data sets, with generally the same characteristics as other non-extended format sequential data sets but with the capability to grow beyond the basic format size limit of 65 535 tracks on each volume. (This is about 3 500 000 000 bytes, depending on the block size.)

Large format data sets reduce the need to use multiple volumes for single data sets, especially very large ones like spool data sets, dumps, logs, and traces. Unlike extended-format data sets, which also support greater than 65 535 tracks per volume, large format data sets are compatible with EXCP and do not need to be SMS-managed. Data sets defined as large format must be accessed using QSAM, BSAM, or EXCP.

**Restriction:** The following types of data sets cannot be allocated as large format data sets:

- ▶ PDS, PDSE, and direct data sets
- ▶ Virtual I/O data sets, password data sets, and system dump data sets

### z/OS V1R9 enhancements

Several commands and services have been updated in z/OS V1R9 to ensure that they can use large format sequential data sets, as follows:

- ▶ Transmit and receive large format data sets > 64 K tracks.
- ▶ Use PRINTDS to print from or to large format data sets.
- ▶ Use REXX EXECIO DISKR/DISRW/DISKRU for REXX I/O to or from large format data sets.
- ▶ Use CLIST OPENFILE/GETFILE/PUTFILE for CLIST I/O to or from large format data sets.
- ▶ Use the REXX LISTDSI function or CLIST LISTDSI statement for gathering size and DSNTYPE for large format data sets.
- ▶ Use large format data sets on the TSO stack

Updates have been made to the following commands and service to ensure that each can handle large format data sets:

- ▶ TSO TRANSMIT, RECEIVE
- ▶ PRINTDS
- ▶ REXX LISTDSI function
- ▶ CLIST LISTDSI statement
- ▶ REXX EXECIO command
- ▶ CLIST OPENFILE/GETFILE/PUTFILE I/O processing

**Note:** Most of the changes are internal and simply provide extended capabilities for TSO/E, REXX, and CLIST users.

### Types of sequential data sets

Large format data sets have the following characteristics:

- ▶ A large format data set is one of the three sequential data set types. This data set might not actually contain a large amount of data. But the fact that it is large format means that it has the capability to grow large in size, beyond 64 K (65535) tracks.



- ▶ A large data set simply refers to one that contains a large amount of data, typically something near the maximum size of a basic format data set.

The three types of sequential data sets are as follows:

- |                        |  |
|------------------------|--|
| <b>Basic format</b>    | A traditional data set, as existed prior to z/OS V1R7. These data sets cannot grow beyond 64 K tracks per volume.  |
| <b>Large format</b>    | A data set (introduced in z/OS V1R7) that has the capability to grow beyond 64 K tracks. The maximum size is x'FFFFFFE' or approximately 16 M tracks per volume.                   |
| <b>Extended format</b> | An extended format data set must be DFSMS-managed. This means that it must have a storage class. These data sets can be striped, and can grow up to x'FFFFFFFE' tracks per volume. |

## 24.2 TSO/E and large format data sets

In releases prior to z/OS V1R9, the TSO/E **TRANSMIT** and **RECEIVE** commands cannot handle large format data sets (DSNTYPE=LARGE) because the 4-byte INMSIZE text unit (key X'102C'), which holds the file size in bytes, is limited to 2 GB.

The function of previous **TRANSMIT/RECEIVE** commands has been extended to the new large data set type.

- ▶ A new TSO/E **TRANSMIT/RECEIVE** text unit, INMLSIZE, allows file size of the transmitted file specified in megabytes (MB).
  - As mentioned, in releases prior to z/OS V1R9, the TSO/E **TRANSMIT** and **RECEIVE** commands could not handle large format data sets (DSNTYPE=LARGE) because the 4-byte INMSIZE text unit, which holds the file size in bytes, is limited to 2 GB.
 

In z/OS V1R9, however, a 4-byte text unit called INMLSIZE is introduced to hold the size of the **TRANSMIT** and **RECEIVE** command operations in MB (up to 4096 TB). If the size of a data set being transmitted is 2 GB or more, INMLSIZE is used instead of INMSIZE.
- ▶ The **TRANSMIT** command has a new **WARN/NOWARN** operand to allow or suppress warning message INMX034I when writing to OUTDA.
 

<b>WARN</b>	<b>WARN</b> means that you can request that the <b>TRANSMIT</b> command issues warning message INMX034I when the warning threshold is initially met, and thereafter whenever the warning interval is met. This is the default if neither <b>WARN</b> nor <b>NOWARN</b> is specified.
<b>NOWARN</b>	<b>NOWARN</b> means that you can request that the <b>TRANSMIT</b> command does not issue warning message INMX034I when the warning threshold is initially met, nor thereafter whenever the warning interval is met.
- ▶ When you use the **TRANSMIT** command for large files, ensure that you have defined sufficient JES spool space.
  - The **TRANSREC** statement's **OUTLIM** operand in IKJTSOxx parmlib member limits the maximum number of records that can be transmitted over the network.

**Note:** The **TRANSMIT** command was also changed to ignore the **OUTLIM** operand when a **TRANSMIT** is directed to a data set (OUTDA).

## TRANSMIT/RECEIVE examples

To create a flat file from a PDS, in a local output data set, the NOWARN parameter indicates that no INMX034I warning messages should be displayed. If the number of records exceeds the max number that is allowed by the OUTLIM operand of the TRANSREC statement in IKJTSOxx parmlib member, the transmission to an output data set will not stop.

```
transmit sc70nje.naidoo da('naidoo.largeds') outda('naidoo.temp.largeds')
nowarn
```

Following is a RECEIVE command to retrieve the file from the flat file created with the previous TRANSMIT command and the messages that occur.

```
RECEIVE INDA('naidoo.temp.largeds')
INMR901I Dataset NAID00.LARGEDS from NAID00 on SC70NJE
INMR906A Enter restore parameters or 'DELETE' or 'END' +
DA('naidoo.new.largeds')
```

## 24.3 TSO PRINTDS command

The TSO **PRINTDS** command allows users to print data sets. You can now use **PRINTDS** to print from or to large format data sets.

In the following example, the **PRINTDS** command is used to write a large format data set to a TODATASET. When the **PRINTDS** command allocates the output data set, it will also be large format.

```
PRINTDS da('naidoo.largeds') todataset('naidoo.image.largeds') notitle
```

**Important:** When PRINTDS writes to an output TODATASET, PRINTDS may be unable to correctly determine the size needed, as when writing a single member of a PDS or PDSE. Or PRINTDS may be unable to allocate the TODATASET, if more than one volume is needed.

In either case, you can pre-allocate the TODATASET of sufficient size and direct the PRINTDS output to the pre-allocated data set.

## 24.4 REXX and CLIST LISTDSI function

You can use the LISTDSI (list data set information) function to retrieve detailed information about a data set's attributes. The attribute information is stored in variables, which can be displayed or used within instructions. The function call is replaced by a function code that indicates whether the call was successful. The LISTDSI function can be used only in REXX execs that run in the TSO/E address space.

The list data set information (LISTDSI) function is very powerful and can provide a significant amount of useful information about an allocated DD or a data set. The REXX LISTDSI function or CLIST LISTDSI statement can now report information about large format data sets. Most of the LISTDSI support is also available in z/OS V1R8.

- ▶ In z/OS V1R9, LISTDSI handles large format data set > 64 K tracks. The following are LISTDSI variables:
  - SYSSEQDSNTYPE returns BASIC, LARGE, EXTENDED.
  - SYSUSED returns the correct size, for < 64 K tracks and > 64 K tracks.

The following REXX exec illustrates using LISTDSI with these variables and lists the returned data set values in parenthesis.

```
/* rexx */
trace n
say 'list large dataset attributes'
call listdsi 'naidoo.large.dataset'
say sysdsname(NAID00.LARGE.DATASET)
say sysvolume(SBOXFH)
say sysdsorg) (PS)
say sysprimary) (70000)
say sysseconds) (15000)
say sysdsms) (SEQ)
say sysunits) (TRACK)
say sysseqdsntype) (LARGE)
say sysused) (183)
exit
```

## 24.5 Enhanced I/O capability in TSO/E for CLIST and REXX

REXX EXECIO reads information from a data set with either the DISKR or DISKRU operands. Using these operands, you can also open a data set without reading its records. EXECIO writes information to a data set with the DISKW operand. In z/OS V1R9, REXX EXECIO has been enhanced to read and write to large format data sets.

Similarly, the CLIST OPENFILE, GETFILE and PUTFILE functions have been enhanced to read and write to large data sets.

## 24.6 Messages related to new support

The following are new and updated messages:

IKJ59042I Terminated. Size of the OUTPUT data set needed exceeds the maximum size that PRINTDS can allocate for a TODATASET.

INMX220I TRANSMIT command terminated. Data set or file being transmitted exceeds maximum size allowed.

INMR908AThe input file attributes are: DSORG=, RECFM=, BLKSIZE=, LRECL=, File size= bytes (File sizes can be specified in K-Bytes or M-Bytes).

INMX034I WARNING: nnn records transmitted. Your installation limit is mmm records.

**Note:** The OUTWARN operand of the TRANSREC statement of the IKJTSOxx parmlib member indicates on what interval the warning message INMX034I should be issued during TRANSMIT. Issuing TRANSMIT with the NOWARN operand suppresses the message.

INMX033I You have EXCEEDED the MAXIMUM TRANSMISSION SIZE set by your installation.

**Note:** With z/OS V1R9, when TRANSMIT is directed to an OUTDA, the OUTLIM maximum is no longer applicable. The TRANSMIT does not terminate when this limit is reached, and message INMX033I is not issued.

However, this limit still applies when transmitting to the NJE network.

## 24.7 Migration and coexistence considerations

The following points should be considered when moving files between different levels of the operating system.

- ▶ If you TRANSMIT a data set greater than 2 GB from z/OS V1R8 to V1R9, TSO/E RECEIVE may get an x37 (out-of-space condition) abend on V1R9 because the V1R8 TRANSMIT cannot send a INMSIZE bigger than x'7FFFFFFF' bytes.
  - The z/OS V1R8 system would be able to send the data set, but with incorrect size.
  - If you pre-allocate a data set of sufficient size on V1R8, you should be able to RECEIVE into it.
- ▶ If you TRANSMIT a data set greater than 2 GB from a z/OS V1R9 system to a V1R8 or earlier, the earlier system will not recognize the new INMLSIZE giving the size in MB (max = x'FFFFFFFF' MB, or approximately 4096 TB).
  - If you pre-allocate a large data set of sufficient size on the z/OS V1R8 system and point the RECEIVE command at it, RECEIVE should be able to receive the V1R9 data.
- ▶ In some situations, TRANSMIT may not accurately determine the size of a data set being transmitted (for example, when transmitting a single member of a PDS or PDSE).
  - Based on the INMSIZE or INMLSIZE sent in the TRANSMIT data, RECEIVE may either not get a data set of sufficient size, or it may get a data set significantly larger than needed.

You can avoid wasting space on the RECEIVE side by pre-allocating the receive data set of sufficient size, and then using the RECEIVE command to receive the data into that data set.



## RMF enhancements

Many different activities are required to keep your system running smoothly, and to provide the best service on the basis of the available resources and workload requirements. Resource Measurement Facility (RMF) is the tool that helps you to perform these tasks. RMF consists of several components that work together in providing the capabilities you need for performance management.

This chapter describes the changes and enhancements made to RMF in z/OS V1R9.

- ▶ RMF enhancements for FICON
- ▶ RMF Monitor III Data Portal
  - Sort capability for full Monitor III reports
- ▶ RMF Spreadsheet Reporter enhancements to simplify performance analysis

## 25.1 RMF enhancements for FICON

The Monitor I/O Queuing Activity report is changed to include two new columns:

- ▶ The estimated Average Number of Open Exchanges
- ▶ The Data Transfer Concurrency within the LCU summary row

The average number of concurrently active I/O operations is provided in the LCU summary line of the Postprocessor I/O Queuing Activity report if at least one FICON channel is connected to the LCU.

**Note:** Open exchanges are the number of I/Os that are concurrently active on a machine.

Figure 25-1 shows the Postprocessor I/O Queuing Activity Report, with two new columns at the end of the report line. In the previous release, a status message appeared indicating that the LCU had been dynamically changed. If a status message is being provided, it is moved now into an extra line immediately following the normal summary line.

I/O QUEUING ACTIVITY																	
z/OS V1R9			SYSTEM ID SC70			DATE 05/28/2007			INTERVAL 30.00.002								
			RPT VERSION V1R9 RMF			TIME 10.00.00			CYCLE 0.100 SECONDS								
TOTAL SAMPLES = 18000		IODF = 05		CR-DATE: 05/23/2007		CR-TIME: 14.00.21		ACT: ACTIVATE									
LCU	CU	DCM MIN	GROUP MAX	CHAN DEF	CHPID PATHS	% DP TAKEN	% CU BUSY	AVG CUB DLY	AVG CMR DLY	CONTENTION RATE	DELAY Q LENG	AVG CSS DLY	HPAV WAIT	MAX	AVG OPEN EXCH	DATA XFER CONC	
0007	C700				8C P	0.542	12.16	0.00	0.0	0.0							
					8D P	0.438	14.72	0.00	0.0	0.0							
					8E NP	0.000	00.00	0.00	0.0	0.0							
					8F NP	0.000	00.00	0.00	0.0	0.0							
		0	0	4		0.000	0.00	0.00	0.0	0.0							
					*	1.816	14.78	0.00	0.0	0.0	0.001	0.00	0.2	0.002	256	0.052	0.001
					*												LCU DYNAMICALLY CHANGED

Figure 25-1 RMF I/O Queuing Activity report

### Field descriptions

RMF uses the connect and disconnect time values to calculate the estimated average number of open exchanges and the data transfer concurrency. Because these counts are currently only available on a device level, device gathering must be active for all devices of those logical control units (LCUs), where the new data should be reported in the I/O Queuing Activity report. Figure 25-2 on page 447 shows the new field descriptions.

Field Heading	Meaning
AVG OPEN EXCH	<p>The Average Number of Open Exchanges on LCU level is estimated as follows:</p> $\text{Avg Open Exch} = \frac{(\text{CMR} + \text{Connect} + \text{Disconnect time} * 128 / 1000 (\text{milliseconds}))}{\text{RMF interval} (\text{milliseconds})}$ <p>where CMR time is the initial command response time</p> <p>All times are in units of 128 microseconds, the RMF interval is in milliseconds</p>
DATA XFER CONC	<p>The Data Transfer Concurrency on LCU level is calculated as:</p> $\text{Data XFER Conc} = \frac{\text{Connect time} * 128 / 1000 (\text{milliseconds})}{\text{RMF interval} (\text{milliseconds})}$ <p>Connect time is in units of 128 microseconds, the RMF interval is in milliseconds</p>

Figure 25-2 New field calculations

### 25.1.1 SMF record changes

SMF 78-3 was changed to include the new fields, as shown in Figure 25-3. The I/O queuing data section is extended by new counts, used to calculate the estimated average number of open exchanges and the data transfer concurrency.

Offsets	Name	Len	Format	Description
3 x3	R783DSTX	1	binary	Data status extension BIT 0 = LCU contains at least one FICON channel BIT 1 = connect time of at least one device invalid BIT 2 = disconnect time of at least one device invalid BIT 3 to 7 = reserved
...	...			...
44 x2C	R783DCTM	4	binary	Accumulated device connect time in units of milliseconds
48 x30	R783DDTM	4	binary	Accumulated device disconnect time in units of milliseconds

Figure 25-3 SMF 78-3 record changes

## 25.2 RMF Monitor III Data Portal

The Resource Measurement Facility (RMF) is an IBM-licensed program that measures selected areas of system activity and presents the data collected in the format of printed reports, System Management Facility (SMF) records, or display reports. RMF is used to evaluate system performance and identify reasons for performance problems. RMF Monitor III is a browser-based interactive monitor that collects data and reports contention for resources and their users. The data allows identification of system bottlenecks and determination of the reasons for possible system performance degradations.

OMEGAMON XE on z/OS provides a launch of the Monitor III data portal from the sysplex Enterprise Overview workspace. You enable the launch by providing the location of the browser on the client host system and the URL of the Monitor III Web interface. You can

configure the launch using the Create or Edit Launch Definitions window in the Tivoli Enterprise Portal, or from a command line by using the **LaunchConfig** command.

The z/OS RMF Distributed Data Server (DDS) provides a Web front-end to sysplex-wide RMF Monitor III online performance data. This Web front-end is called the RMF Monitor III Data Portal. By using a Web browser such as Mozilla 1.4 or above, Netscape 7.0 or above, or Microsoft Internet Explorer 5.5 or above, it is possible to display XML documents with XSL style sheets. With the browser, there is instant access to more than 600 z/OS performance metrics.

You must start DDS as follows:

```
F RMF,DDS
```

From a Web browser, enter the following:

```
http://<hostname>:8803
```

Figure 25-4 shows the first screen that appears on the browser.

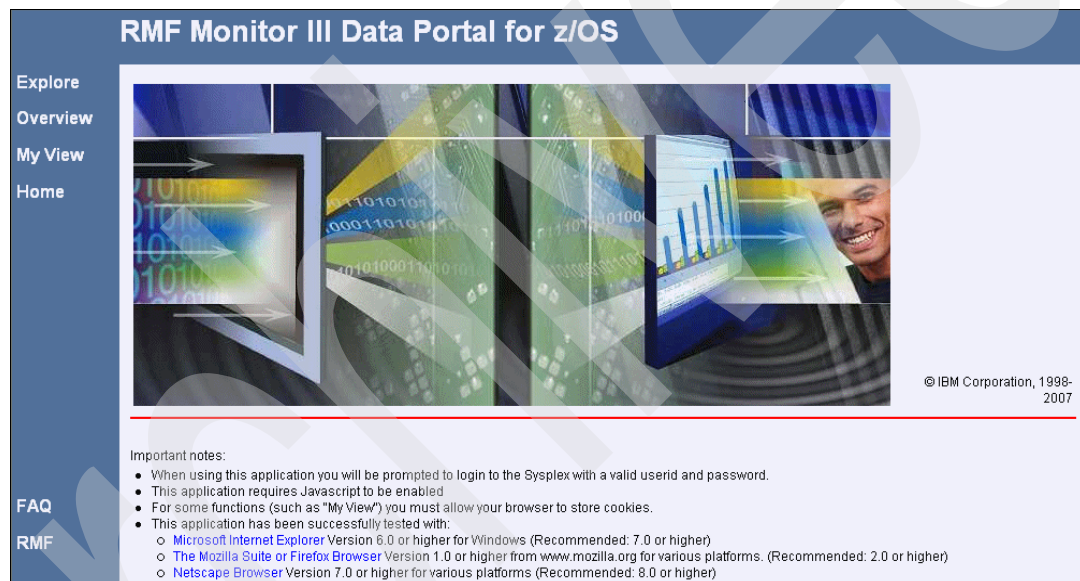


Figure 25-4 RMF Monitor III Data Portal

The RMF Monitor III Data Portal uses the same data model as the RMF PM Java client. The same set of resources and metrics are available. Basically, a resource can be monitored by the selection of one or more metrics from the corresponding metrics list. For a comprehensive analysis, you might possibly associate a complete Monitor III report with a specific resource.

The Monitor III sysplex-wide reports have been assigned to the sysplex resource and the Monitor III single system reports have been assigned to the MVS\_IMAGE resource.

- ▶ Monitor III reports assigned to the sysplex are:
  - CACHDET, CACHSUM, CFACT, CFOVER, CFSYS, and SYSSUM
- ▶ Monitor III reports assigned to an MVS\_IMAGE are:
  - CHANNEL, CPC DELAY, DEV, DEVR, DSND, ENCLAVE, IOQ, OPD, PROC, PROCU, STOR, STORC, STORCR, STORF, STORS, and SYSINFO



## 25.2.1 Sort capability for full Monitor III reports

With RMF for z/OS V1R9, RMF Monitor III Data Portal is enhanced to fully support the sorting of Monitor III reports. You can select any column to sort the values in this column in either ascending or descending order. Reports which consist of blocks of lines logically belonging together will keep these blocks as one while sorting is executed.

After you have navigated to the resource you are interested in (in the Data Portal), select the report containing the desired performance information. In our example, we use the Coupling Facility activity (CFACT) report on the SANDBOX sysplex. The CFACT report will then be displayed in your browser. Figure 25-5 shows the lines in the table in the same default sorting order as they would appear in the corresponding ISPF Monitor III report.

RMF Monitor III Data Portal for z/OS

Explore  
Overview  
My View  
Home  
FAQ  
RMF

RMF Report [.,SANDBOX,SYSPLEX] : CFACT (Coupling Facility Activity)  
Time Range: 05/29/2007 11:57:00 - 05/29/2007 11:58:00

Structure Name	Structure Type	Structure Status	Extended Structure Status	System Name	CF Utilization %	Structure Execution %	Sync Rate	Sync Avg Service Time	Sync Request Count	Async Rate	Async Avg Service Time	Async Request Count
DB8FU_LOCK1	LOCK	A	ActivePersistent	*ALL	2.7	0.0	0.1	385	1	4.0	2064	240
	LOCK			SC63			0.0	0	0	2.0	2053	120
	LOCK			SC64			0.1	385	1	2.0	2075	120
	LOCK			SC65			0.0	0	0	0.0	0	0
	LOCK			SC70			0.0	0	0	0.0	0	0
DB8FU_SCA	LIST	A	ActivePersistent	*ALL	5.9	0.0	0.0	0	0	3.9	1735	232
	LIST			SC63			0.0	0	0	1.9	1651	116
	LIST			SC64			0.0	0	0	1.9	1820	116
	LIST			SC65			0.0	0	0	0.0	0	0
	LIST			SC70			0.0	0	0	0.0	0	0
EJESGDS_WTSCPLX4	LIST	A	ActivePersistent	*ALL	0.2	0.0	0.0	0	0	0.0	0	0
	LIST			SC63			0.0	0	0	0.0	0	0

Figure 25-5 RMF CFACT report

To sort the CF Utilization % column, simply click the column header as shown in Figure 25-6 on page 450. Two arrows are displayed to visualize the sort order (descending, in our example). In addition, the color of the column which was used as the sort criterion changes to yellow to indicate this currently active sort criterion.

**Note:** A second click on the column header will reverse the sort order to ascending in our current example.

**RMF Monitor III Data Portal for z/OS**

Explore  
Overview  
My View  
Home

← → ↩ 20070529115700

RMF Report [,SANDBOX,SYSPLEX] : CFACT (Coupling Facility Activity)

Time Range: 05/29/2007 11:57:00 - 05/29/2007 11:58:00

Structure Name	Structure Type	Structure Status	Extended Structure Status	System Name	CF Utilization %	Structure Execution %	Sync Rate	Sync Avg Service Time	Sync Request Count	Async Rate	Async Avg Service Time	Async Request Count
IGWLOCK00	LOCK	A	ActivePersistent	*ALL	43.4	0.0	0.0	0	0	0.0	0	0
SYSZWLM_991E2094	CACHE	A	ActiveInUse	*ALL	20.3	0.0	0.0	0	0	1.8	4090	109
IXC_DEFAULT_4	LIST	A	ActiveInUse	*ALL	18.0	0.0	0.0	0	0	0.5	2257	29
RRS_RMDATA_1	LIST	A	ActiveInUse	*ALL	9.6	0.0	0.1	667	2	10.0	1662	597
ISGLOCK	LOCK	A	ActiveInUse	*ALL	9.4	0.0	3.4	313	202	13.9	1488	834
IXC_DEFAULT_2	LIST	A	ActiveInUse	*ALL	7.9	0.0	0.0	0	0	0.4	2981	24
DB8FU_SCA	LIST	A	ActivePersistent	*ALL	5.9	0.0	0.0	0	0	3.9	1735	232
IXC_DEFAULT_3	LIST	A	ActiveInUse	*ALL	4.5	0.0	0.0	0	0	0.7	2387	40
DB8FU_LOCK1	LOCK	A	ActivePersistent	*ALL	2.7	0.0	0.1	385	1	4.0	2064	240
RRS_MAINUR_1	LIST	A	ActiveInUse	*ALL	1.7	0.0	0.1	757	1	0.6	2058	34
SYSTEM_OPERLOG	LIST	AP	ActiveDuplexPrimary	*ALL	1.5	0.0	0.0	0	0	0.1	3721	3

FAQ  
RMF

Figure 25-6 RMF CFACT report sort capability

In cases where a single line does not contain all necessary information within itself, the sorting needs some special treatment which takes the dependency on the row's neighbors into account. Monitor III reports which require this special treatment are:

► Sysplex reports

**CFACT**

The first column of the table represents the CF structure name. Each CF structure section consists of multiple lines which would lose their context through generic sorting. The sort procedure will keep these structure blocks together and sort according to highest (for descending sort order) or lowest (for ascending sort order) value within the block.

**CFSYS**

The first column of the table represents the CF name. It follows the same argument and solution as in the CFACT report, keeping blocks together.

**SYSSUM**

The first column of the table represents the WLM entity name (group or class). The report will be sorted line by line, filling in the WLM entity name in the first column where this information is missing.

► System reports

**CPC**

Sorting is done only within an area with the same processor type. The lines marked with \*CP, \*ICF and PHYSICAL are summary lines which will be kept at the beginning or end of each block during the sort.

**DELAY**

The first column of the table represents a job name. The rows marked with an asterisk in front of the job name are summary lines except for \*MASTER\*. After sorting, the report will consist of two sections. The first section contains all summary lines. The second section will contain all the remaining lines.

**DEVR**

This report contains blocks of lines that belong together and must be sorted as one item.

**IOQ**

This report shows rows with values for a single CHPID (marked ,C' in the last column), summary lines for DCM-managed CHPIDs belonging to the same LCU (marked ,D' in the last column) and summary lines for all

CHPIDs belonging to an LCU (marked 'S' in the last column). The sort procedure will keep these blocks with the same CHPID and LCU together and sort according to highest (for descending sort order) or lowest (for ascending sort order) value within the block.

- STORC:** The report will be sorted like the DELAY report. Summary lines start with an asterisk (\*) and also with a percent (%) sign.
- STORS:** The report will be sorted like the SYSSUM report (filling in missing information into the first column).
- SYSINFO:** This report will be sorted like the DELAY report (building two blocks of sorted lines). Additionally, missing information will be filled into the first column like in SYSSUM report.

## 25.3 SpreadSheet Reporter enhancements

The RMF Spreadsheet Reporter is the powerful workstation solution for graphical presentation of RMF Postprocessor data. It is used to convert your RMF data to spreadsheet format and generate representative charts for all performance-relevant areas.

RMF Spreadsheet Reporter enhancements include:

- ▶ New RMF Spreadsheet options
- ▶ zAAP and zIIP support
- ▶ Report Class Periods
- ▶ RMF XCF Activity Report
- ▶ Process user-defined overview records

### 25.3.1 New RMF Spreadsheet options

The Spreadsheet Reporter GUI is enhanced to offer two new Report Options:

- ▶ Workload Activity (Report Classes)
  - Enable this option to create a Workload Activity RMF Postprocessor Report with Report Class and Report Class Period information.
  - When set, the following postprocessor option is used.
    - SYRPTS(WLMGL(RCLASS,RCPER,WGROUP,POLICY))
- ▶ XCF Activity
  - Enable this option to create an XCF Activity RMF Postprocessor Report.
  - When set, the following postprocessor option REPORTS(XCF) is used.

The new Spreadsheet Reporter options are shown in Figure 25-7 on page 452.

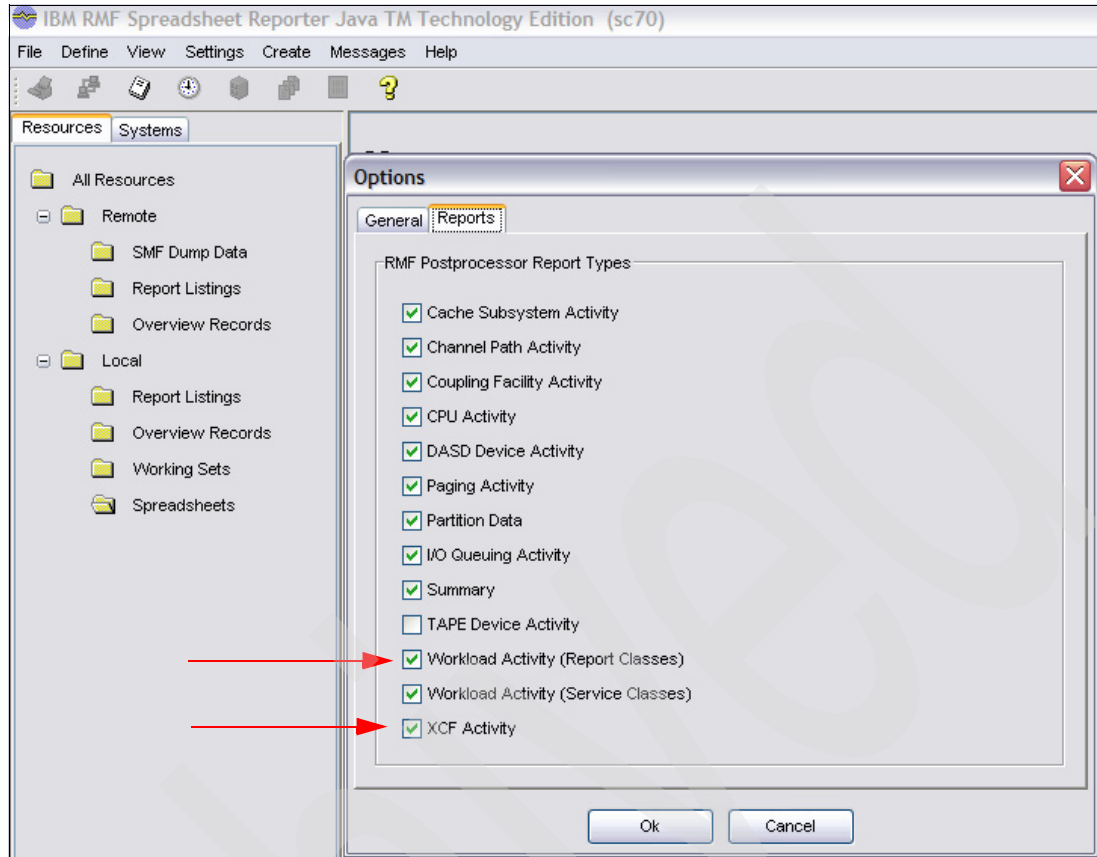


Figure 25-7 Spreadsheet Reporter options

### 25.3.2 zAAP and zIIP support

The Workload Activity report spreadsheet is based on the RMF Workload Activity Postprocessor Report. The spreadsheet is enhanced to support the new zAAP and zIIP metrics.

The RepExcD sheet, as shown in Figure 25-8 on page 453, displays Using and Delay samples for selected Workloads. In addition, zIIP and zAAP Using and Delay samples are also available.

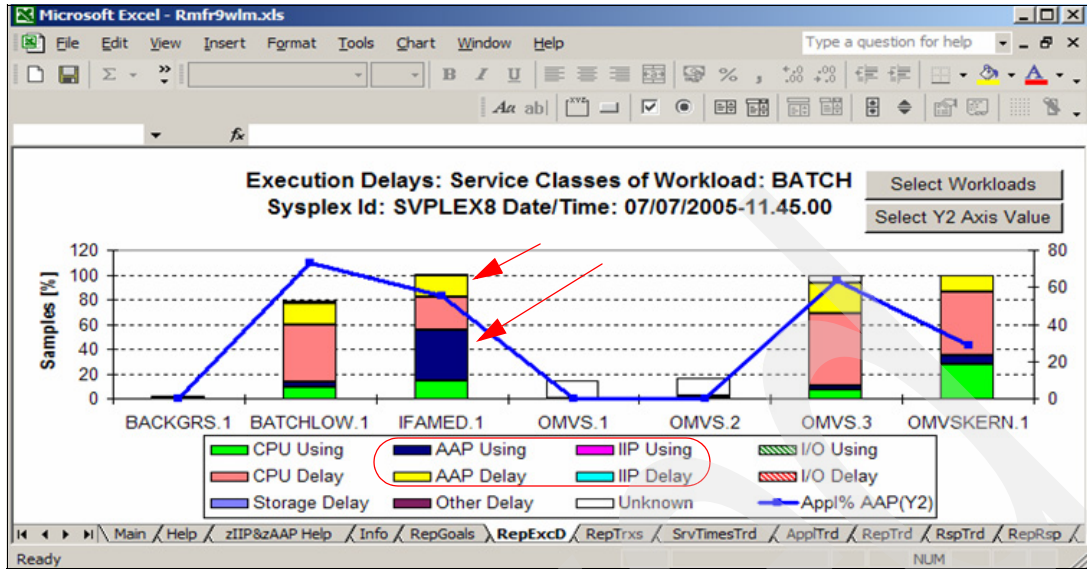


Figure 25-8 RepExcd report

The ApplTrd sheet, shown in Figure 25-9, reports application execution time for a selected workload as a trend chart.

The following metrics are reported:

- ▶ APPL % CP
- ▶ APPL % AAP
- ▶ APPL % AAP on CP
- ▶ APPL % IIP
- ▶ APPL % IIP on CP

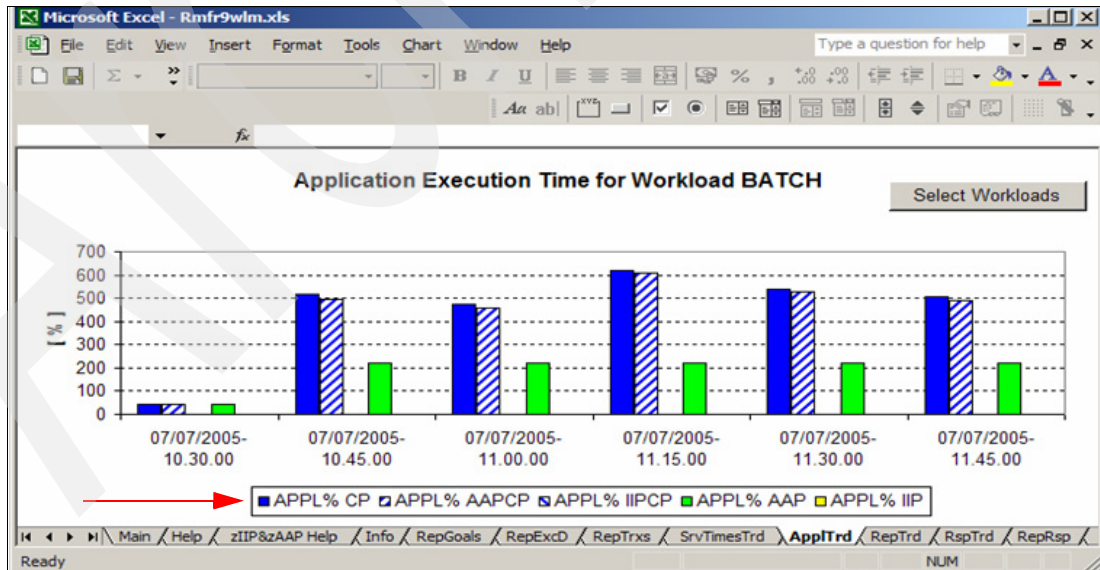


Figure 25-9 ApplTrd report

The LPAR Trend report spreadsheet is based on the RMF Partition Data Postprocessor Report. The spreadsheet is enhanced to support the new zAAP and zIIP processor types and metrics.

As shown in Figure 25-10, on the LparInt sheet for each partition the related CP, AAP, IIP, ICF, IFL and IIP performance data is reported. The user can select the Interval which is reported, and also select the metric which is displayed on the chart.

The available metrics for the charts are:

- ▶ Logical Effective Dispatch Time %
- ▶ Physical Effective Dispatch Time %
- ▶ # Logical Processors
- ▶ Actual Weight
- ▶ Actual Weight %

**Note:** The new processor types are also honored on the LparType, OneLparTrd, LparsTrd and RepWeight sheets.

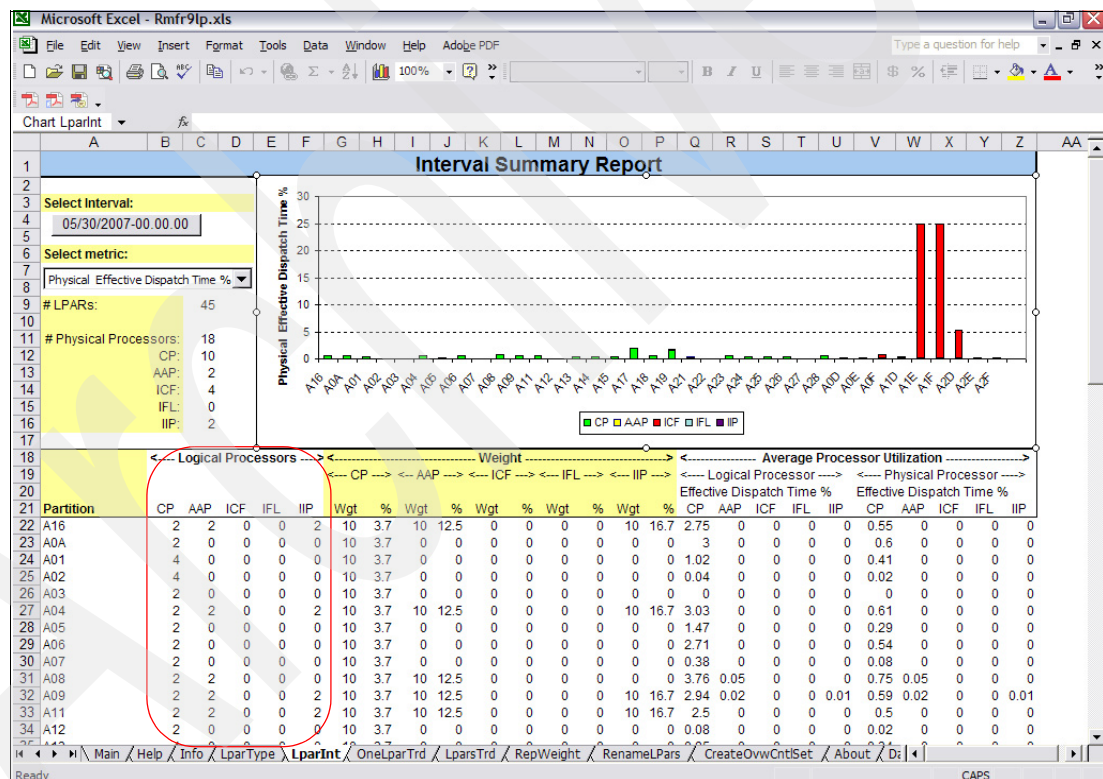


Figure 25-10 Interval Summary report

### 25.3.3 Report Class periods

The Workload Activity report spreadsheet is based on the RMF Workload Activity Postprocessor Report. The spreadsheet is enhanced to support Report Class Periods.

As shown in Figure 25-11 on page 455, below the listed service classes and service class periods you will find now the report class and report class periods in the list.



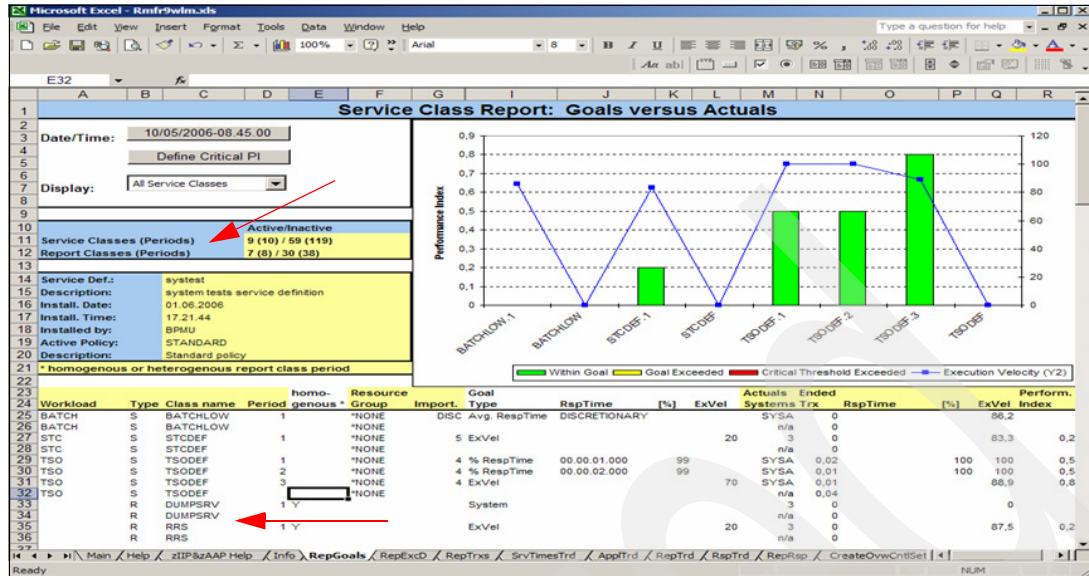


Figure 25-11 RepGoals report

### 25.3.4 RMF XCF Activity Report

The new XCF Activity report spreadsheet macro is based on the RMF XCF Activity Postprocessor Report. You can combine interval reports from several systems to create a sysplex performance view.

In the macro you will find several sheets for XCF Transport Class and XCF Path performance metrics. The following three sheets report XCF Transport Class performance data:

- ▶ TCOverview:
  - This provides an overview of the Transport Class Buffer Size.
- ▶ TCSystems:
  - Based on the XCF Usage by System Section of the XCF Activity Report, several selectable Outbound Request metrics are reported.
  - You can reduce the content with a filter for: Sending System, Receiving System, and Transport Class.
  - If you set the Sending System filter to \*All Systems, the spreadsheet will summarize the data of the selected system reports, generating a sysplex view. If you just select one sending system, a single system view is generated.
  - Additionally, you can specify the reporting category for which the data is reported:
    - by Sending System
    - by Receiving System
    - by Transport Class
- ▶ TCBuffers
  - Based on the XCF Usage by System Section of the XCF Activity Report, this chart reports the distribution of the Outbound Requests for the categories: SML, FIT, BIG (without OVR), BIG (with OVR).
  - Additionally, a selectable metric is reported on the secondary Y axis.
  - You can set a filter to specify the Sending System, Receiving System, and Transport class.

Figure 25-12 shows a XCF Outbound Request Summary report and the Filter and the Metric selection scroll-down fields.

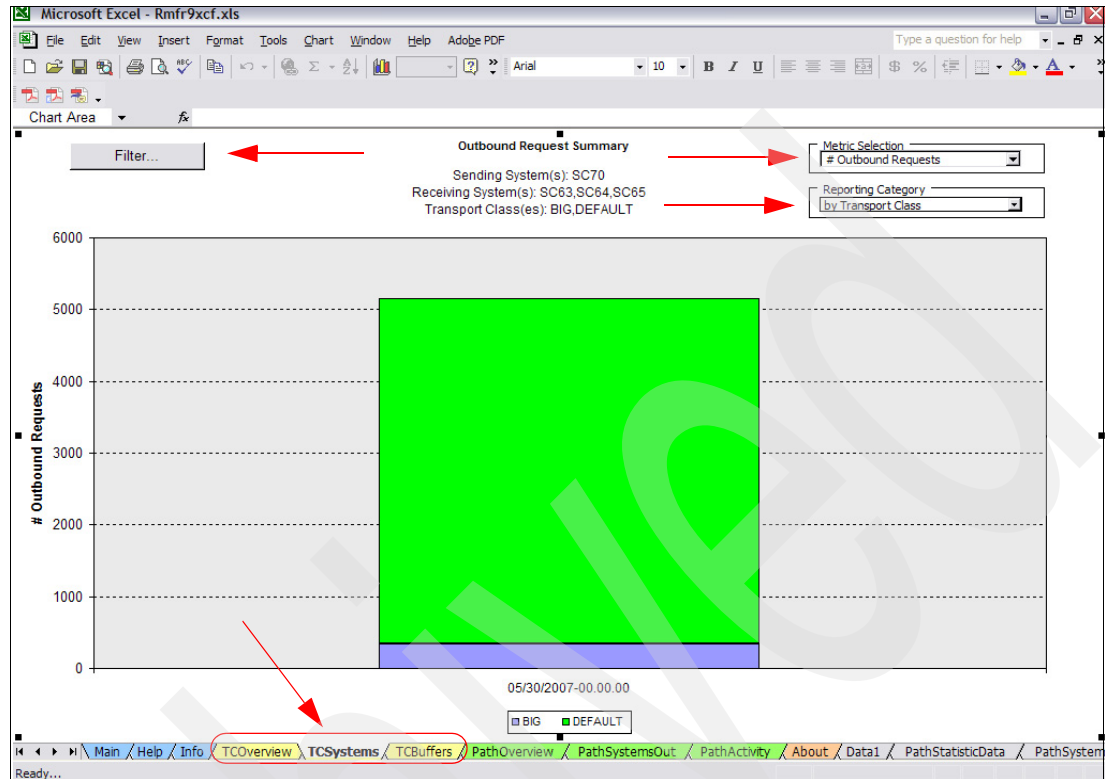


Figure 25-12 XCF Activity report

You can also set a filter to specify the Sending System, Receiving System, Transport class, and Path. Based on the XCF Path Statistic Section of the XCF Activity Report, the chart in Figure 25-13 on page 457 reports the distribution of the Outbound Requests for the following categories:

- ▶ Path Available
- ▶ Busy and Retries

Additionally, a selectable metric is reported on the secondary Y axis. A filter can be used to specify the Sending System, Receiving System, Transport class, and Path.



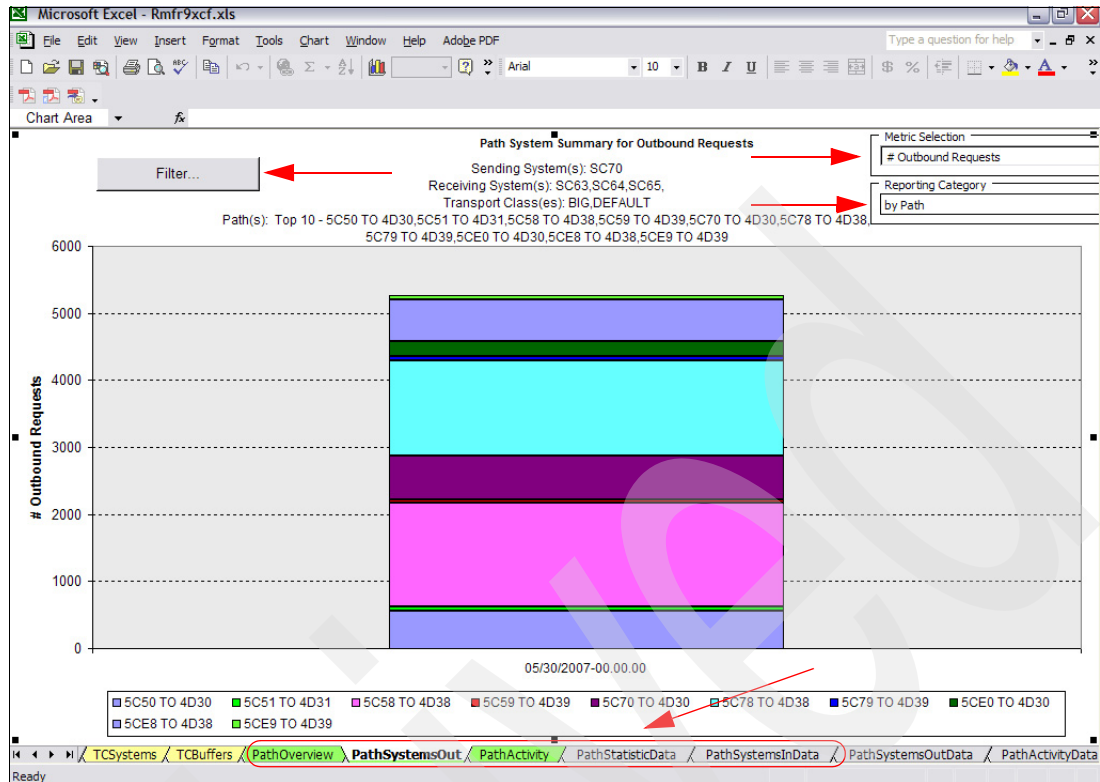


Figure 25-13 PathSystemsOut report

### 25.3.5 Process user-defined overview records

The new Overview Report spreadsheet is able to process any working set, based on user-defined overview records. It automatically creates charts from the data. Additionally, the user can customize the charts. On the setup sheet, the user can customize the chart titles and the metric description.

The following charts are generated:

**IntervalChart** Data from one interval is reported. The user can select the interval that should be reported. Additionally, the user can specify the metrics that are reported in the charts (a subset from the available data).

**DayChart** Data from one day is reported. The user can select the day that should be reported. It is also possible to select the chart type: line chart, stacked array chart, or array chart (depending on the metrics of the user-defined overview record). Additionally, the user can specify the metrics that are reported in the charts (a subset from the available data).

**TrendChart** Data is reported as a trend chart. The user can select the chart type from a line chart, stacked array chart, or array chart (depending on the metrics of the user-defined overview record). Additionally, the user can specify the metrics that are reported in the charts (a subset from the available data).

Figure 25-14 on page 458 shows a DayChart. When you select Chart Options, it shows you only a subset of the available data (for example, only OCPU1 and OCPU2).

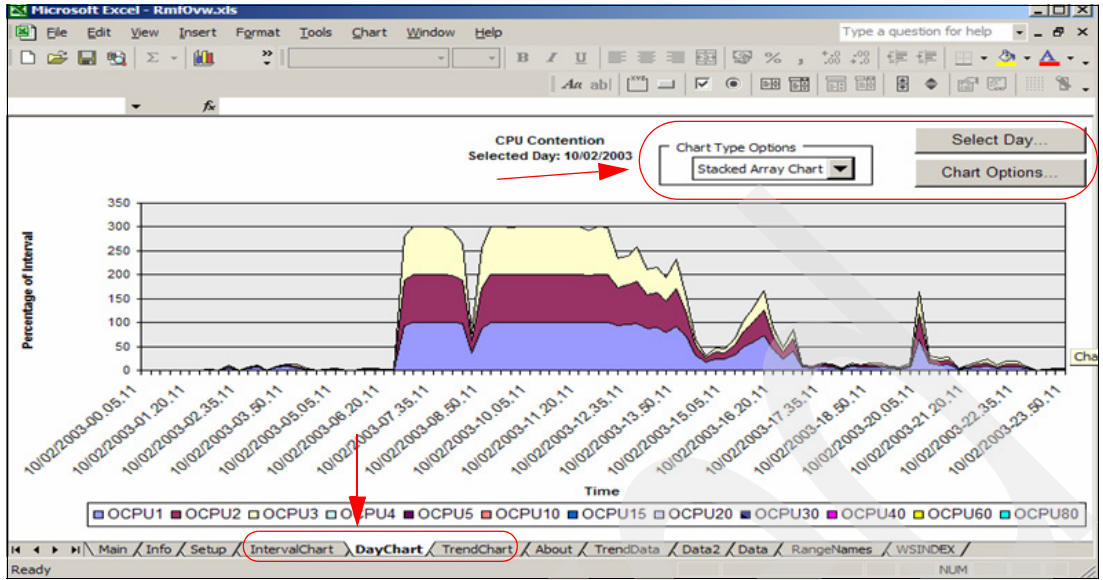


Figure 25-14 DayChart report

## XML enhancements

z/OS XML System Services (z/OS XML), a BCP component introduced in z/OS V1R8, is a system-level XML parser intended for use by system components, middleware, and applications that need a simple, efficient, non-validating XML parser. It provides a buffer-in, buffer-out processing model in which the document to parse is provided by the caller in one buffer, and the z/OS XML parser creates a parsed record stream in another buffer, also provided by the caller. Arbitrarily large documents can be processed by replenishing the input buffer with more data and reusing the output buffer or specifying a new one when it is filled.

z/OS XML can natively handle the following character set encodings: UTF-8, UTF-16 (big endian), EBCDIC-1047, and EBCDIC-037. z/OS XML is invoked as a callable service and provides an assembler interface for callers to use.

It is accessible from normal environments such as batch and UNIX System Services, as well as from more esoteric environments such as SRB mode and cross-memory.

This chapter describes the modifications for XML system services made in z/OS V1R9.

- ▶ Improved performance of XML System Services parser
- ▶ C/C++ APIs
- ▶ Enabling the parser to run on zAAP specialty engines

## 26.1 XML System Services

DB2 was the primary driver of the design requirements, used for their expanded XML features. The reasons for implementing the parser are:

- ▶ To provide an XML parser that is part of the BCP and requires no extra installation or downloading. This allows other z/OS components to exploit the function, and makes the function always available for use by customers and middleware.
- ▶ To meet the requirements for z/OS-specific environments, such as X-memory mode or SRB mode. XML System Services runs in key and state of the caller, and supports all z/OS operating environments.
- ▶ To address the version and service-level issues that the z/OS XML Toolkit has experienced with multiple exploiters.
- ▶ To have new z/OS features dependent on a separate installable package.
- ▶ Open software APIs are often not stable. The APIs change and are deprecated. With z/OS XML System Services, it is a typical BCP component, with support of downward compatibility, as is usually expected with z/OS.
- ▶ To exploit the widespread availability of C and C++ language development and performance tools that such applications can take advantage of. Many programs are already written in C/C++ and will no longer need to switch into PL/X or assembler to invoke XML System Services, or make inter-language calls without the benefit of C/C++ header files.

## 26.2 Performance improvements

The combination of C/C++ APIs and zAAP specialty engine support makes z/OS attractive for applications that require non-validating XML parsing. This will be very useful for IBM and external (that is, ISV and customer) exploiters because assembler and PL/X skills are becoming increasingly rare; there is widespread availability of C language development and performance tools that such applications can take advantage of; and many programs are already written in C/C++ and will no longer need to switch into PL/X or assembler to invoke XML System Services or make inter-language calls without the benefit of C/C++ header files.

**Note:** The Metal is a new option on the C compiler. The Metal compiler option, which is provided with z/OS V1R9, generates code that does not have Language Environment run-time dependencies. In addition, language features are provided to embed small pieces of HLASM source within C statements.

There are mostly changes to the internals to improve performance. There is approximately a 30% improvement of cycles per bytes parsed with z/OS V1R9 over previous releases.

Some feature changes are not available on a control call, such as recovery and jobsteps that own storage features. A control reset enhancement was requested by DB2, but may be used by any caller of XML System Services.

Features that can be changed on the control call include the following:

- ▶ XEC\_FEAT\_STRIP\_COMMENTS
  - This effectively strips comments from the document by not returning any comments in the parsed data stream. Default: off.

- ▶ XEC\_FEAT\_TOKENIZE\_WHITESPACE
  - This sets the default token value for white space preceding markup in the root element to an explicit white space value. Default: off; white space is returned as character data.
- ▶ XEC\_FEAT\_CDATA\_AS\_CHARDATA
  - This returns CDATA in records with a CHARDATA token type. The content of these records may contain text that would normally have to be escaped to avoid being handled as markup. Default: off.

## 26.3 C/C++ APIs

The combination of C/C++ APIs and zAAP specialty engine support makes z/OS attractive for applications that require non-validating XML parsing. This will be very useful for IBM and external (that is, ISV and customer) exploiters because assembler and PL/X skills are becoming increasingly rare; there is widespread availability of C language development and performance tools that such applications can take advantage of; and many programs are already written in C/C++ and will no longer need to switch into PL/X or assembler to invoke XML System Services or make inter-language calls without the benefit of C/C++ header files.

Table 26-1 lists the functions available to invoke XML C/C++ APIs.

**Note:** The C/C++ APIs are also available to z/OS 1.6 and 1.7 with APAR OA18713.

Table 26-1 C/C++ APIs for XML

C/C++ API	Function
gxlpControl	<p>This is a general purpose service which provides control functions for interacting with the z/OS XML parser. The function performed is selected by setting the <code>ctl_option</code> parameter using the constants defined in <code>gxlhxec.h</code>. These functions include:</p> <p>GXLHXEC_CTL_FIN</p> <p>The caller has finished parsing the document. Reset the necessary structures so that the PIMA can be reused on a subsequent parse, and return any useful information about the current parse.</p> <p>GXLHXEC_CTL_FEAT</p> <p>The caller wants to change the feature flags. A XEC_CTL_FIN function will be done implicitly.</p>
gxlpInit	<p>This callable service initializes the PIMA and records the addresses of the caller's system service routines (if any). The PIMA storage is divided into the areas that will be used by the z/OS XML parser to process the input buffer and produce the parsed data stream.</p>
gxlpParse	<p>This callable service parses a buffer of XML text and places the result in an output buffer.</p>
gxlpQuery	<p>This service allows a caller to obtain the XML characteristics of a document. The XML characteristics are either the default values, the values contained in an XML declaration, or a combination of both.</p>

C/C++ API	Function
gxlpTerminate	The gxlpTerminate callable service releases all resources obtained (including storage) by the z/OS XML parser and resets the PIMA so that it can be reinitialized or freed.

### What is a PIMA

In addition to control information, the PIMA is used as a memory area to store temporary data required during the parse. When the z/OS XML parser needs more storage than was provided in the PIMA, additional storage is allocated. Everything that the z/OS XML parser needs to complete the parse of a document is kept in the PIMA, and any associated memory extensions that parser may allocate during the parse process.

The caller must also provide input and output buffers on each call to gxlpInit. In the event that either the text in the input buffer is consumed, or the parsed data stream fills the output buffer, the z/OS XML parser will return XRC\_WARNING, along with a reason code indicating which buffer (possibly both) needs the caller's attention. It also indicates the current location, and the number of bytes remaining in each buffer, by updating the buffer\_addr and buffer\_bytes\_left parameters passed on the parse request.

This process is referred to as *buffer spanning*. If the entire document has been processed when the z/OS XML parser returns to the caller, the parse is complete and the caller proceeds accordingly. If the caller needs to parse another document, it will have to call gxlpInit again to either completely reinitialize the existing PIMA area or initialize a new PIMA area from scratch.

Another option is to use the finish/reset function of the gxlpControl z/OS XML parser service to reset the PIMA so that it can be reused. This is a lighter-weight operation that preserves certain information that can be reused across parsing operations for multiple documents. This improves the performance for subsequent parses, because this information can be reused instead of being rebuilt from scratch. Reusing the PIMA in this way is particularly beneficial to callers that need to handle multiple documents that use the same symbols (for example, namespaces and local names for elements and attributes). The PIMA can only be reused in this way when the XML documents are in the same encoding and the same z/OS XML parser options are used.

### 26.3.1 Sample project

To demonstrate how to use the new introduced APIs a small program is created. The C source is shown in Figure 26-1 on page 463. The program first initializes the XML System services using gxlpInit. Next, a sample is shown using XML Services to parse with the function gxlpParse. At the end, the buffers are removed and a release of the PIMA is done using the gxlpControl. To terminate XML Services completely, use the gxlpTerminate function.

```

* function      : invoke the XML System Services      *
* history      : 24.05.2007 Lutz      first creation  *
#include <gxlhtml.h>      /* include XML Services      */
                        /* located in /usr/lib or      */
                        /* /usr/lib/Metal            */
                        /* SYS1.SIEANDRV.H          */

#include <stdlib.h>
#include <stdio.h>
int main(){
int rc, rsn;              /* return and reason codes */
void * PIMA;             /* Parse Instance Memory Area */
long lPIMA = GXLHXEC_MIN_PIMA_SIZE; /*
int ccsid = GXLHXEC_ENC_IBM_1047; /* set Coded Character Set ID */
int feature_flags;      /* which parser features used */
void * inputBuffer;     /* define input area */
void * outputBuffer;    /* define output area */
void * outputBufferStart; /* define inputarea start point */
long * ctl_data;        /*
void * CTL_DATA_PTR;    /*
long input_buffer_len = 0; /* initialize */
long output_buffer_len = 0; /* some */
int option_flags = 0;    /* variables */
int i;                  /* loop counter */
int *array;
CTL_DATA_PTR = &ctl_data; /* pointer to control data */
struct _GXLHXSV testing_struct;
testing_struct.XSV_COUNT = 0; /* no exits passed in */
/*-----*/
/* start main program */
/*-----*/

printf("starting test program for XML System Services\n");
PIMA = (void*) malloc(GXLHXEC_MIN_PIMA_SIZE);
if(PIMA == NULL) return -1; /* allocate PIMA */
inputBuffer = (void *) malloc(1000); /* allocate input buffer */
/* allocate output buffer */
outputBufferStart = outputBuffer = (void*) malloc(4000);
feature_flags = 0; /* no special features */
/* invoke gxlpInit to initialize environment */
gxlpInit(PIMA,lPIMA,ccsid,0,testing_struct,NULL,&rc,&rsn);
printf("gxlpInit: rc = %d rsn = %x\n", rc, rsn);
/* fill input buffer for gxlpParse */
sprintf((char *) inputBuffer,"<?xml version=\"1.0\" encoding=\"IBM-1047\"?> <a> /a>");
input_buffer_len = 49;
output_buffer_len = 4000;
/* invoke gxlpParse */
gxlpParse(PIMA, &option_flags,&inputBuffer, &input_buffer_len, &outputBuffer, &output_buffer_len,
&rc, &rsn);
printf("gxlpParse: rc = %d rsn = %x\n", rc, rsn);
array = (int *) outputBufferStart;
/* print the result of gxlpParse */
for (i = 0; i < 30; i++)
printf("%8.8x ", array[i]);
printf("\n");
/* finishing parsing and reset PIMA */
gxlpControl(PIMA,GXLHXEC_CTL_FIN,CTL_DATA_PTR,&rc,&rsn);
printf("gxlpControl: rc = %d rsn = %x\n", rc, rsn);
/* release all storage and release PIMA */
gxlpTerminate(PIMA, &rc, &rsn);
printf("gxlpTerminate: rc = %d rsn = %x\n", rc, rsn);
free(PIMA); /* free */
free(inputBuffer); /* allocated */
free(outputBuffer); /* buffers */
return 0; }

```

Figure 26-1 Sample C program for using XML

## 26.3.2 How to compile

To create the needed load module, you use the regular C compiler shipped with z/OS. Figure 26-2 shows a sample call of the C compiler for a 64-bit environment.

```
cxx -Wc,lp64,expo,"langlvl(extended)" -c -o xmltest64.o -I/u/lutz xml.c
cxx -Wl,lp64 -o xmltest64 -L/u/lutz xmltest64.o /usr/lib/gxlxml4.x
```

Figure 26-2 Invoke Compiler for 64-bit

Figure 26-3 shows the same, but with the XP-LINK option.

```
cxx -Wc,xplink,expo,"langlvl(extended)" -c -o xmltestxp.o -I/u/lutz xml.c
cxx -Wl,xplink -o xmltestxp -L/u/lutz xmltestxp.o /usr/lib/gxlxml1.x
```

Figure 26-3 Invoke compiler with XP-LINK

**Important:** You can use the same source code for both 31-bit and 64-bit load modules. You only use different compiler/linker options and a separate import library.

To create a MVS load module, use the job shown in Figure 26-4.

```
//XMLTEST JOB , 'COMPILE', NOTIFY=LUTZ, REGION=512M,
// CLASS=A, MSGCLASS=X, MSGLEVEL=(1,1), TIME=1440
/*JOBPARM S=SC70
//COMP EXEC EDCCL,
// INFILE='LUTZ.SOURCE(XMLTEST)',
// OUTFILE='LUTZ.LOADLIB(XMLTEST), DISP=SHR',
// CPARAM='XPLINK, EXPO, "LANGLVL(EXTENDED)"',
// LPARM='MAP'
//COMPILE.SYSLIB DD DSN=SYS1.SIEAHDV.H, DISP=SHR
//LKED.SYSLIB DD DSN=CEE.SCEELKED, DISP=SHR
// DD DSN=CEE.SCEELKEX, DISP=SHR
// DD DSN=CEE.SCEEBND2, DISP=SHR
//XMLINCL DD DSN=SYS1.SIEALNKE, DISP=SHR
//LKED.SYSIN DD *
INCLUDE XMLINCL(GXLCXML1)
NAME XMLTEST(R)
/*
//
```

Figure 26-4 Compile in batch

Compiling the source program requires a new include library called gxlxml.h. This is provided by z/OS in the libraries listed in Table 26-2.

Table 26-2 Compile and link environments

Environment	Compiler headers	Link library
z/OS UNIX	/usr/include	/usr/lib/gxlxml1.x XP-LINK /usr/lib/gxlxml4.x 64-bit



Environment	Compiler headers	Link library
z/OS UNIX with Metal option	/usr/include/Metal	/usr/lib/gxlxml1.x XP-LINK /usr/lib/gxlxml4.x 64-bit
z/OS	SYS1.SIEAHDRV.H	SYS1.SIEALNKE SYS1.SIEASID side decks

After compilation, you need to link the load module. Therefore you must include the XML System Services link stub called `gxlxml1.x` (for XP-LINK) or `gxlxml4.x` (for 64-bit applications). Figure 26-5 shows the output of our sample program.

```
LUTZ:/u/lutz: >./xmltestxp
starting test program for XML System Services
gxlpInit:      rc = 0 rsn = 0
gxlpParse:     rc = 4 rsn = 1301
f00f0000 00000020 00000000 00000000 00000000 00000064 00000000 00000000 f01f0000
0000001f 00000003 f14bf000 000008c9 c2d460f1 f0f4f700 000000f0 2f000000 00001500
00000181 00000000 00000000 f07f8000 00000010 00000004 4061816e 00000000 00000000
00000000 00000000 00000000
gxlpControl:  rc = 0 rsn = 1301
gxlpTerminate: rc = 0 rsn = 1301
LUTZ:/u/lutz: >
```

Figure 26-5 Output of the sample program

### 26.3.3 zAAP considerations

z/OS XML System Services provides the ability for parsing operations to be run on a zSeries Application Assist Processor (zAAP). The z/OS XML parser, when executing in TCB mode, is eligible to run on a zAAP in environments where one or more zAAPs are configured. Ancillary z/OS XML System Services, such as the query service and the control service, as well as the StringID exit and memory management exits, are not eligible to run on a zAAP.

Execution of z/OS XML System Services parsing operations on a zAAP occurs transparently to the calling application. The XML code provided with z/OS V1R9 switches automatically over to zAAP specialty engines prior to parsing a document if there is at least one zAAP processor available in the system. There is currently no way to prevent this behavior. There is no switch done to zAAP when running in SRB mode.

**Note:** The zAAP support for XML System Services is also available to z/OS 1.6 and 1.7 with APAR OA20308.

To determine whether your system has a zAAP installed or not, use the **DISPLAY M=CPU** z/OS system command. Figure 26-6 on page 466 is an example of this command; it shows there are two zAAP processors installed (02 and)3) and eligible XML work.

```

ISF031I CONSOLE LUTZ ACTIVATED
D M=CPU
IEE174I 15.06.38 DISPLAY M 412
PROCESSOR STATUS
ID CPU SERIAL
00 + 16991E2094
01 + 16991E2094
02 +A 16991E2094
03 +A 16991E2094
04 +I 16991E2094
05 +I 16991E2094

CPC ND = 002094.S18.IBM.02.00000002991E
CPC SI = 2094.710.IBM.02.00000000002991E
CPC ID = 00
CPC NAME = SCZP101
LP NAME = A16 LP ID = 16
CSS ID = 1
MIF ID = 6

+ ONLINE - OFFLINE . DOES NOT EXIST W WLM-MANAGED
N NOT AVAILABLE

A APPLICATION ASSIST PROCESSOR (zAAP)
I INTEGRATED INFORMATION PROCESSOR (zIIP)
CPC ND CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID CENTRAL PROCESSING COMPLEX IDENTIFIER
CPC NAME CENTRAL PROCESSING COMPLEX NAME
LP NAME LOGICAL PARTITION NAME
LP ID LOGICAL PARTITION IDENTIFIER
CSS ID CHANNEL SUBSYSTEM IDENTIFIER
MIF ID MULTIPLE IMAGE FACILITY IMAGE IDENTIFIER

```

Figure 26-6 Display of installed CPs and zAAPs



## RRS enhancements

Resource Recovery Services (RRS) provides a set of services that implement the two-phase commit protocol on the z/OS platform. Your resource manager follows the two-phase commit protocol to protect resources by invoking these services and providing exit routines.

This chapter provides information about the new functions in ATRRRS with z/OS V1R9. The following new functions are described:

- ▶ ATRQSRV batch support
- ▶ Resource manager unregister

## 27.1 ATRQSRV batch support

RRS has support for displaying RRS information via ISPF panels. Figure 27-1 shows a sample ISPF RM detail information panel. In order to obtain any information about RRS resources, it was necessary to navigate through multiple panels; there was no way to display information using batch.

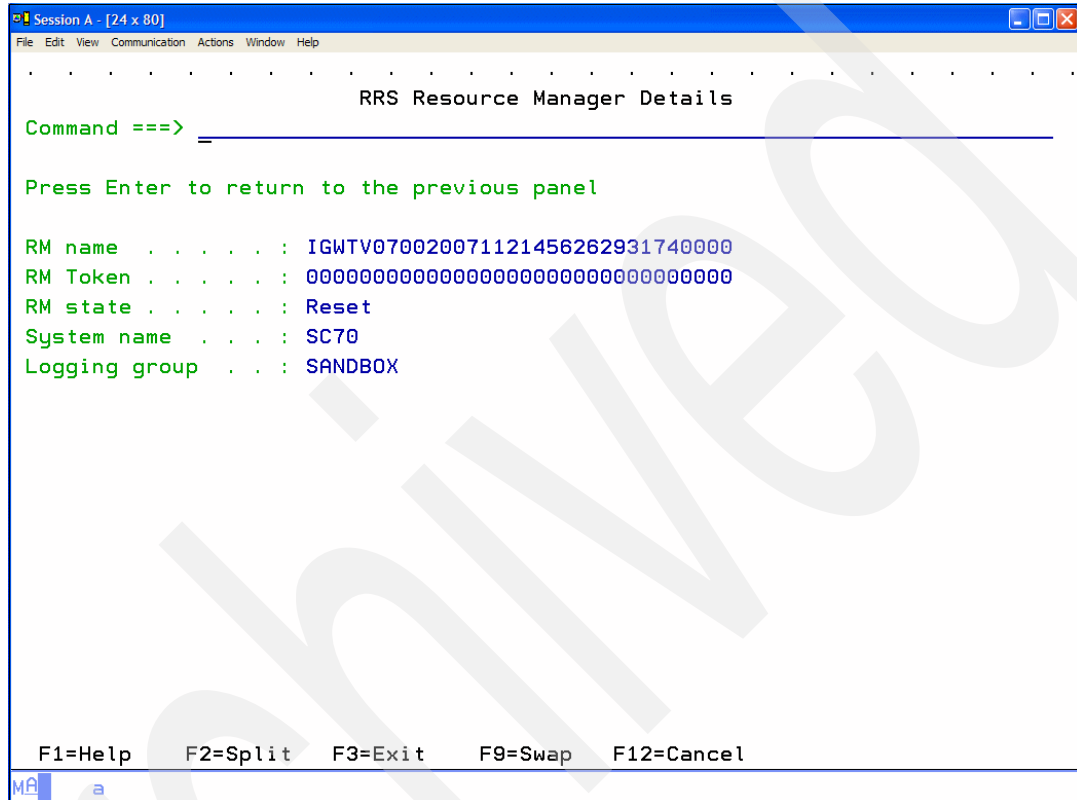


Figure 27-1 RM detail ISPF panel

### 27.1.1 ATRQSRV utility

RRS information can be especially useful in problem determination. Another method for capturing RRS information is by using the ATRQSRV batch utility. Figure 27-2 shows a sample job for obtaining RRS RM information.

```
//LUTZRRS JOB , 'KUEHNER', NOTIFY=&SYSUID,  
// CLASS=A, MSGCLASS=X, MSGLEVEL=(1,1), TIME=1440  
//LISTATR EXEC PGM=ATRQSRV  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
RMINFO RMNAME(IGWTV070020071121456262931740000)  
/*  
//
```

Figure 27-2 Sample JCL for ATRQSRV use

## SYSIN DD statement

The SYSIN DD statement defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member in a partitioned data set or PDSE.

The SYSIN DD statement is required for each use of ATRQSRV. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified for this block size.

You can pass several control statements to the ATRQSRV via the SYSIN DD statement, as shown in Table 27-1. It is also possible to pass more than one control statement to the ATRQSRV program.

Table 27-1 Control statements given to ATRQSRV

Statement	Meaning
LOGINFO	Browse the Archive, Main/Delayed UR, Restart and RM Data log streams
URINFO	Query unit of recovery (UR) information
RMINFO	Query resource manager (RM) information
WMINFO	Query work manager (WM) information
SYSINFO	Query RRS sysplex and logging group information
REMOVINT	Remove interest(s) from URs
COMMIT	Force an InDoubt transaction to the COMMIT state
BACKOUT	Force an InDoubt transaction to the BACKOUT state
DELETERM	Delete a resource manager from RRS
UNREGRM	Unregister RM to clean up the a resource manager's involvement with RRS

Table 27-2 lists the possible return codes returned by the ATRQSRV program.

Table 27-2 Possible return codes given by ATRQSRV

Return Code	Description
0	Service completed successfully
4	Errors detected. Messages appear in SYSPRINT DD.
8	Errors writing to the SYSPRINT DD.
12	Errors opening SYSPRINT DD.
16	Errors closing SYSPRINT DD.
4095	Unexpected error.

Figure 27-3 on page 470 shows the output of the sample batch job.



In either case, RRS performs the RM unregister processing to mark all UR interests for the failing RM as failed and unset the RM's RRS exits, as well as the RM unregistration processing with the registration services. If RRS experiences an internal failure before completing the RM unset processing, RM could be left in an unregistered state with registration services but was still set with RRS. This situation cannot currently be resolved without recycling RRS and its resource managers, which is undesirable. These means that prior to z/OS 1.8, you have to cancel the RRS address space. Starting with z/OS 1.8, you can use the SETRRS CANCEL function to recycle the RRS address space.

Currently there are three ways to unregister RMs:

- ▶ RRS ISPF panels
- ▶ Applications via the updated ATRSRV interface
- ▶ JCL via the ATRQSRV batch utility

In the process of restarting, a resource manager will call the CRGSEIF, Set\_Exit\_Information, service to notify RRS of its intent to work with RRS.

When the resource manager is still set with RRS, CRGSEIF will return a code of 8004, ATR\_RM\_ACTIVE\_ON\_ANOTHER\_SYSTEM, meaning that the resource manager currently has exits set. Any of the ATR services, such as ATRIBRS, Begin\_Restart, would fail with a 701 return code ATR\_RM\_STATE\_ERROR, meaning that the resource manager is not in set state. Most likely the resource manager is in RUN state. From the RRS ISPF panels Option 2, Display Resource Manager information, use the **UNREGISTER RM** command to unregister the resource manager with RRS.

**Tip:** The command will only work if the resource manager is unregistered with registration services, but still registered with RRS.

### **New ISPF panels**

Figure 27-5 on page 472 shows the new Unregister RM command in the commands list.

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
RRS Resource Manager List Row 1 to 3 of 3
Command ==> _____ Scroll ==> PAGE

Commands: v-View Details u-View URs r-Remove Interest d-Delete RM
          n-Unregister RM

S  RM Name                               State      System   Logging Group
_  IGWTV070020071121456262931740000  Reset     SC70    SANDBOX
_  IGWTV070020071181152224736540000  Reset     SC70    SANDBOX
_  IGWTV070020071201503218732890000  Reset     SC70    SANDBOX
***** Bottom of data *****

F1=Help      F2=Split    F3=Exit     F5=ListErr  F6=Refresh  F7=Up
F8=Down      F9=Swap     F12=Cancel

MA a

```

Figure 27-5 New ISPF Unregister RM command

After you type the unregister action character to the desired resource and press Enter, you receive a confirmation panel similar to the one shown in Figure 27-6 on page 473.



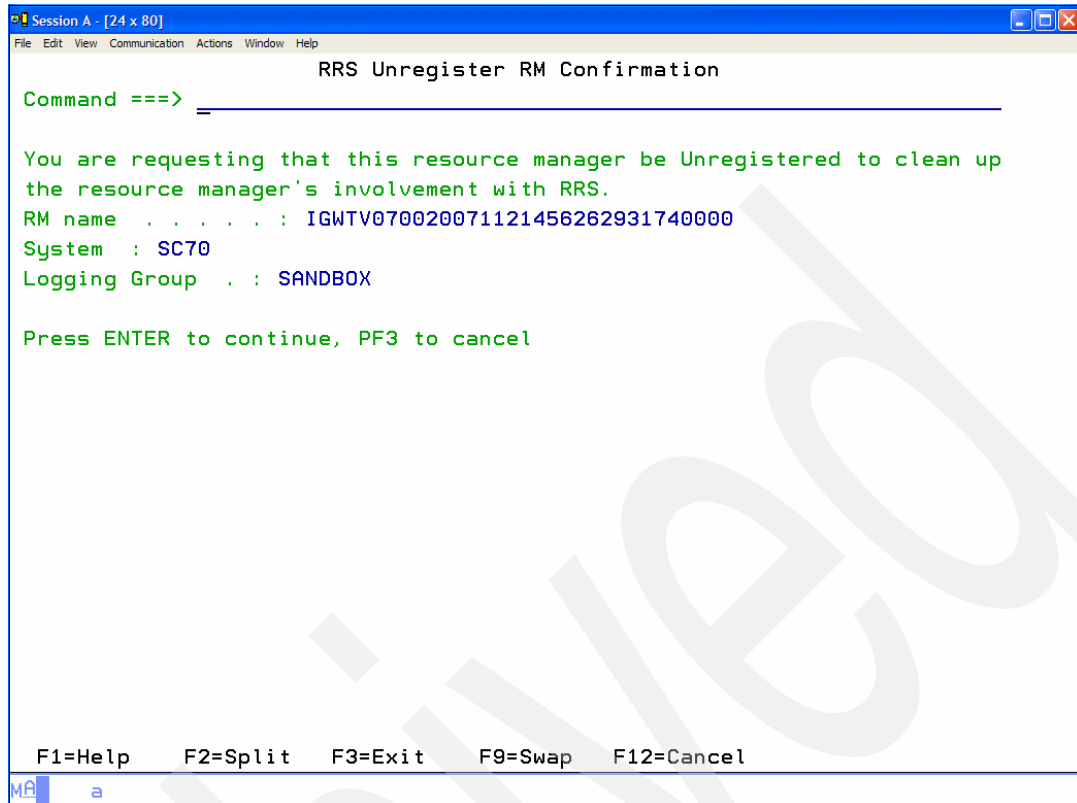


Figure 27-6 Confirmation dialog

After confirmation, and assuming the RM is unregistered with registration services, the RM is unregistered with RRS and new message ATR534I is displayed to indicate a successful unregister. The RM is left on the panel but its state changed to reset. The RM can now restart.

In some cases, the RM may not be valid for unregister; for example:

- ▶ If the new message ATR536I is displayed following the unregister request, it means that the RM selected for unregister is still registered with registration services and cannot be unregistered with RRS.
- ▶ If the new message ATR537I is displayed following the unregister request, it means that a resource manager in the reset or unset state is already considered unregistered with RRS, so it cannot be unregistered again.

### Updated ATRSRV interface

The ?ATRSRV callable service has been extended with an additional keyword REQUEST(UNREGRM, as shown in Figure 27-7 on page 474.

```

?ATRSRV  + REQUEST(REMOVINT)
          + REQUEST(COMMIT)
          + REQUEST(BACKOUT)
          + REQUEST(REMOVRM)
          + REQUEST(UNREGRM) [RMNAME(xrmname)]
            [GNAME({xgname | current_gname})]
            [SYSNAME({xsysname | current_sysname})]
            [RCTABLE({xrctable | 0})
              RCNUM({xrcnum})]
          RETCODE = retcode
          RSNCODE = rsncode

```

Figure 27-7 Modified ATRSRV service

**Restriction:** If the service is routed to a non z/OS V1R9 system, the following message is issued:

ATR538I The ATRSRV request was processed on a downlevel RRS system that could not honor the request.

# Language Environment

Language Environment (LE) provides a common run-time environment for IBM versions of certain high level languages (HLLs), namely C, C++, COBOL, Fortran, and PL/I, in which you can run existing applications written in previous versions of these languages as well as in the current Language Environment-conforming versions. Before Language Environment, each of the HLLs had to provide a separate run-time environment.

Language Environment combines essential and commonly used run-time services (such as routines for run-time message handling, condition handling, storage management, date and time services, and math functions) and makes them available through a set of interfaces that are consistent across programming languages. With Language Environment, you can use one run-time environment for your applications, regardless of the application's programming language or system resource needs because most system dependencies have been removed. Language Environment provides compatible support for existing HLL applications; most existing single-language applications can run under Language Environment without being recompiled or relink-edited. POSIX-conforming C applications can use all Language Environment services.

This chapter describes the following enhancements:

- ▶ `inconv()` enhancements
- ▶ Enhancements of CEEDUMP
- ▶ Enhancements of IPCS BERBX LEDATA
- ▶ `edcmtext`
- ▶ `fdlibm` replacement
- ▶ HEAP pools performance improvement
- ▶ ASCII locale support and Turkish locale update.
- ▶ z/OS UNIX support for CEEBLDTX
- ▶ CLER run-time option change support
- ▶ CEELRR ACTION=INIT,XPLINK=YES
- ▶ AMODE 64 CELQPIPI Service Vector (LOAD/DELETE)
- ▶ XPLINK enhancements

## 28.1 iconv() enhancements

The iconv() family of functions supports conversion of a sequence of characters encoded in one character set to a sequence of characters in another character set. Similar support is available from Unicode Services. Continued standalone support of conversion services via the iconv() family of functions is redundant. Additionally, Unicode Services supports conversion of many more to/from character sets than iconv().

The iconv() family of functions has been modified to utilize Unicode Services Conversion Services under the covers. The modifications are transparent to existing applications. The iconv() family function interfaces remain unchanged for existing conversions except where noted in the migration section of this document. Thousands of additional to/from character set conversions are now supported.

**Important:** As mentioned, these changes are transparent to existing applications.

Two new environment variables and four new errno values have been introduced and are described here.

### New error codes

Following are the new error codes:

<b>ECUNNOENV</b>	A CUN_RS_NO_UNI_ENV error was issued by Unicode Services. Action: Refer to Support for Unicode: Unicode Services documentation for user action.
<b>ECUNNOCONV</b>	A CUN_RS_NO_CONVERSION error was issued by Unicode Services Action: Refer to Support for Unicode: Unicode Services documentation for user action.
<b>ECUNNOTALIGNED</b>	A CUN_RS_TABLE_NOT_ALIGNED error was issued by Unicode Services. Action: Refer to Support for Unicode: Unicode Services documentation for user action.
<b>ECUNERR</b>	iconv() encountered an unexpected error while using Unicode Services. Refer to message EDC6258 for additional information. Message EDC6258 will indicate an error code and reason code issued from Unicode Services which can be used by users to search Unicode: Unicode Services documentation for additional user action.

### New environment variables

The following environment variables are new in z/OS V1R9:

#### ► **\_ICONV\_MODE**

The **\_ICONV\_MODE** environmental variable selects the behavior mode for the iconv\_open(), iconv(), and iconv\_close() family of functions. Table 28-1 on page 477 lists the **\_ICONV\_MODE** values that can be set.

Table 28-1 `_ICONV_MODE` values

Value	Meaning
C	A user has created its own <code>iconv()</code> converter. Use the user-created converter first. If the user-created converter is not located, then use Unicode Services to perform the conversion.
U	The user uses Unicode conversion services to perform all conversions. This is the default value for <code>_ICONV_MODE</code> .

### ► `_ICONV_TECHNIQUE`

This variable tells Unicode conversion services which conversion technique to use during conversions. The default value is `LMREC`. All possible values are listed in Table 28-2.

Table 28-2 `_ICONV_TECHNIQUE` values

Value	Meaning
R	Roundtrip conversion. Roundtrip conversions between two CCSIDs assure that all characters making the roundtrip arrive as they were originally.
E	Enforced Subset conversion. Enforced Subset conversions map only those characters from one CCSID to another that have a corresponding character in the second CCSID. All other characters are replaced by a substitution character.
C	Customized conversion. Customized conversions use conversion tables that have been created to address some special requirements.
L	Language Environment behavior conversion. Environment behavior conversions use tables that map characters like the <code>iconv()</code> function of the Language Environment Runtime library does. These conversions differ from others primarily in their mapping of the EBCDIC newline (NL) character to ASCII and the Unicode linefeed (LF) character.
M	Modified Language Environment behavior conversion. Modified Language Environment behavior conversions use tables that map characters as the <code>iconv()</code> function of the Language Environment Runtime library does for converters ending with C (for example, IBM-932C).
0-9	User-defined conversions. User-defined conversions are supported. For more information, refer to Appendix C, "Defining CCSIDs and conversion tables" in the publication <i>Support for Unicode: Unicode Services</i> , document number SA22-7649.

**Note:** `ICONV_UCS2` and `_ICONV_PREFIX` environment variables will be honored while searching for user-created converters. They will not be honored while using Unicode Services.

## 28.1.1 Migration actions

The `iconv()` users who have created their own conversion tables and want the `iconv()` family of functions to use them will need to set the `_ICONV_MODE` environment variable to `C`. The `_ICONV_UCS2` and `_ICONV_PREFIX` environment variables do not have any meaning when using Unicode Services. The `Iconv()` function returns the number of nonidentical conversions performed during a conversion.

Beginning in z/OSV1R9, a more strict interpretation of a nonidentical conversion will be used by `iconv()`. Thus it is possible that the nonidentical conversion count will be higher in z/OSV1R9 than in previous releases when converting the same input buffer contents.

In pre-z/OSV1R9 releases, `iconv()` used either a single byte or a double-byte substitution character in mixed (converters containing both single and multibyte data) character set conversions, never both. In z/OSV1R9, `iconv()` will use a single byte substitution character when converting single-byte characters and a multibyte substitution character when converting multibyte characters in a mixed character set conversion.

## 28.2 CEEDUMP enhancement

The following enhancements were made for dump processing:

- ▶ There is a new runtime option to control characteristics of a CEEDUMP data set.
- ▶ There is an enhanced traceback section of a Language Environment dump.
- ▶ There is an enhanced condition information section of a Language Environment dump.
- ▶ Job information is added to the Language Environment dump page header.
- ▶ New messages identify the start and end of a Language Environment dump.
- ▶ Language Environment dump processing is suppressed when the CEEDUMP ddname is defined as dummy.

### CEEDUMP run-time option

The CEEDUMP run-time option is used to specify options to control the processing of the Language Environment dump report. It is now possible to control the Language Environment dump processing. Figure 28-1 shows the syntax of the new CEEDUMP option.

```
CEEDUMP( pageLen,  
        SYSOUT=*|SYSOUT=class|SYSOUT=(class,,form),  
        FREE=END|FREE=CLOSE,  
        SPIN=UNALLOC|SPIN=NO)
```

Figure 28-1 CEEDUMP control statement

The default setting for CEEDUMP is for all environments (non-CICS, CICS, AMODE64).

```
CEEDUMP=( (60,SYSOUT=*,FREE=END,SPIN=UNALLOC),OVR)
```

Table 28-3 on page 479 lists detailed descriptions of all supported CEEDUMP options. Note that some of these options are ignored in some environments.

The `SYSOUT=`, `FREE=`, and `SPIN=` suboptions do not have any effect on a CEEDUMP report taken under CICS. If a CEEDUMP DD card is explicitly coded in a job step, Language Environment ignores any `SYSOUT` class, form-name, `FREE`, or `SPIN` specified in the CEEDUMP run-time. The `SYSOUT=class` suboption is overridden by `_CEE_DMPTARG` when this environment variable is used at the same time to indicate the `SYSOUT` class. The `page_len` suboption is overridden by the `CEE3DMP PAGESIZE` option.

Table 28-3 CEEDUMP option list

Option	Description
<b>pagelen</b>	<ul style="list-style-type: none"> <li>▶ Default value is 60.</li> <li>▶ Valid values are 0, and 10 through 999999999.</li> <li>▶ A value of 0 indicates that there should be no page breaks in the dump report.</li> <li>▶ This option can be overridden by the pagesize parameter on the call to the CEE3DMP callable service.</li> </ul>
<b>SYSOUT</b>	<p>Class:</p> <ul style="list-style-type: none"> <li>▶ Default class is *</li> <li>▶ Valid values for class are A through Z, 0 through 9 or *</li> <li>▶ This option can be overridden by _CEE_DMPTARG if this environment variable is used to indicate the sysout class.</li> </ul> <p>Form:</p> <ul style="list-style-type: none"> <li>▶ There is no default form name.</li> <li>▶ Valid values for form-name are made up of 1 to 4 alphanumeric or national (\$,#,@) characters according to JCL rules.</li> </ul>
<b>FREE</b>	<ul style="list-style-type: none"> <li>▶ FREE=END – CEEDUMP data set is unallocated at the end of the last step that references the data set.</li> <li>▶ FREE=CLOSE - CEEDUMP data set is unallocated when it is closed.</li> </ul>
<b>SPIN</b>	<ul style="list-style-type: none"> <li>▶ SPIN=UNALLOC – CEEDUMP data set is available for processing immediately when it is unallocated.</li> <li>▶ SPIN=NO – CEEDUMP data set is available for processing as a part of the output at the end of the job, regardless of when the data set is unallocated.</li> </ul>

**Note:** When you set CEEDUMP to DUMMY inside your JCL, then no CEE dump will be produced during an abend.

### Migration considerations

Users of CEEUOPT or CELQUOPT created with HLE7740 and link-edited with an application that will run on a older release of Language Environment may have run-time options ignored without notification. In this case, users should install APAR PK29028 on down-level systems.

## 28.2.1 Enhanced traceback section

In prior releases, Traceback was only one section. Very long program unit names were truncated. The names of executables in the z/OS UNIX file system were not shown. This caused problems during diagnoses of LE dumps. Figure 28-2 shows an example of a traceback taken from a z/OS V1R8 system.

Traceback:											
DSA Addr	Program	Unit	PU Addr	PU Offset	Entry	E Addr	E Offset	Statement	Load Mod	Service	Status
20A43E80	///	POSIX.CRTL.C(CELDLL)									
			20DC83B0	+0000012E	dump_n_perc	20DC83B0	+0000012E	34	CELDLL	1.3.B	Call
20A43DC8			0CD0F628	-00000106	CEEPGTFN	0CD0F4C8	+0000005A		CEEPLPKA		Call
20A40CA8	CEEHDSP		0CC55518	+000024D0	CEEHDSP	0CC55518	+000024D0		CEEPLPKA	UK20433	Call
20A40208	///	POSIX.CRTL.C(CELSAMP)									
			20A26478	+000009BA	main	20A26478	+000009BA	150	*PATHNAM	1.1.D	Exception
20A400F0			0E284606	+000000C2	EDCZMINV	0E284606	+000000C2		CEEEV003		Call
20A40030	CEEBEXT		0CC25418	+000001B6	CEEBEXT	0CC25418	+000001B6		CEEPLPKA	HLE7730	Call

Figure 28-2 Traceback before V1R9

Figure 28-3 shows the traceback from the same dump, but taken on a z/OS V1R9 system. The report is more flexible and more easily readable, and is similar to what is currently produced for AMODE 64 programs. The compile date and compile attributes columns are new. The Fully Qualified Names section displays long program unit and pathnames of executables in the z/OS UNIX file system.

```

Traceback:
DSA  Entry      E  Offset  Statement  Load Mod      Program Unit      Service  Status
1    dump_n_perc +0000012E 34      CELDLL      CELDLL         CELDLL           1.3.B    Call
2    CEEPGTFN    +0000005A      CEEPLPKA
3    CEEHDSP     +000024BC      CEEPLPKA      CEEHDSP        D1908     Call
4    main       +000009BA 150     ce1samp.exe   CELSAMP        1.1.D    Exception
5    EDCZMINV    +000000C2      CEEEV003
6    CEEBBEXT    +000001B6      CEEPLPKA      CEEBBEXT       D1908     Call

DSA  DSA Addr  E  Addr  PU Addr  PU Offset  Comp Date  Compile Attributes
1    210F6E80 2147B3B0 2147B3B0 +0000012E 20070105  C/C++     POSIX EBCDIC  HFP
2    210F6DC8 20BA7EB8 20BA8018 -00000106 20061215  LIBRARY   POSIX
3    210F3CA8 20AEC068 20AEC068 +000024BC 20061215  CEL       POSIX
4    210F3208 20A26478 20A26478 +000009BA 20070105  C/C++     POSIX EBCDIC  HFP
5    210F3F00 20F930EE 20F930EE +000000C2 20061215  LIBRARY   POSIX
6    210F3030 20ABADB8 20ABADB8 +000001B6 20061215  CEL       POSIX

Fully Qualified Names
DSA  Entry      Program Unit      Load Module
1    dump_n_perc  //'POSIX.CRTL.C(CELDLL)'  CELDLL
4    main        //'POSIX.CRTL.C(CELSAMP)'  ./ce1samp.exe

```

Figure 28-3 Traceback in V1R9

The Traceback produced by the IPCS Verbexit LEDATA is also changed. However, it does not have the Fully Qualified Names section and never displays statement numbers.

### Enhanced condition information section

The possible bad branch location only appears for 0C1 and 0C4 abends when the Entry offset is zero or negative. Only a small portion of the GPREG STORAGE section is shown, but the section does include storage around all 16 registers. Figure 28-4 shows example output of a 0C1 system abend. The new, added information is shown in bold text, and it provides more useful information to use for diagnostic purposes.

```

Condition Information for Active Routines
Condition Information for (DSA address 20FCB2B0)
CIB Address: 20FCBC70
Current Condition:
  CEE0198S The termination of a thread was signaled due to an unhandled condition.
Original Condition:
  CEE3201S The system detected an operation exception (System Completion Code=0C1).
Location:
  Program Unit: Entry: funca Statement: Offset: -20900978
Possible Bad Branch: Statement: Offset: +0000005A
Machine State:
ILC..... 0002      Interruption Code..... 0001
PSW..... 078D1400 80000002
GPR0..... 20FCB350 GPR1..... 20FCB2A0 GPR2..... 20FCB2A0 GPR3..... 209009B2
GPR4..... A09A0BBC GPR5..... 20912648 GPR6..... 20900AA4 GPR7..... 20900098
GPR8..... 00000030 GPR9..... 80000000 GPR10.... A0E699E2 GPR11.... A09A0AD8
GPR12.... 209139B0 GPR13.... 20FCB2B0 GPR14.... A09009D4 GPR15.... 00000000

Storage dump near condition, beginning at location: 00000000
+000000 00000000 Inaccessible storage.
GPREG STORAGE:
Storage around GPRO (20FCB350)
-0020 20FCB330 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 |.....|
+0000 20FCB350 0808CEE1 20FCB2B0 20FCE470 A09D6B3A A09EFFD8 20FCB350 20FCB7A8 20912648 |.....U.....Q...&...y.j..|

```

Figure 28-4 Enhanced condition information



## Additional job information

New page headers in the Language Environment dump provide additional information about the causing job or z/OS UNIX-related information. Table 28-4 lists the additional information that is added to the dump headlines listed by the calling environment.

Table 28-4 Added fields list

Calling environment	Added fields information
BATCH POSIX(ON)	ASID Job ID Job name Step name PID Parent PID User name
BATCH POSIX(OFF)	ASID Job ID Job name Step name UserID
z/OS UNIX shell	ASID PID Parent PID User name
CICS	Task Number Transaction ID

## New messages for dump status

Two new messages are introduced in z/OS V1R9. The first message indicates the beginning of the dump taken. The second message is shown after the dump is taken by the system. Figure 28-5 displays an example of these new messages for dump processing.

```
CEE3845I CEEDUMP processing started.  
CEE3846I CEEDUMP processing completed
```

Figure 28-5 New dump processing messages

This mechanism might help you to separate multiple LE dumps produced by a single job.

## 28.3 edcmtext utility

Because of the reuse of the same errno values in multiple failure paths, both within a function and cross-function, it is sometimes difficult to identify the cause of a failure. An additional error indicator was needed to help identify the specific failure. This support will help both customers and IBM support personnel to diagnose application problems.

The solution was to provide errno2 support in the z/OS XL C/C++ Run-Time Library, similar to the errno2 support provided in z/OS UNIX System Services. The goal of the errno2 work for the C run-time library is to produce a unique errno2 value for each failure path where errno is also set. Each errno2 value will be used only once. The errno2 values themselves will not be programming interfaces.

However, a new utility, `edcmtext`, is provided that can be used to query the description of the failure and the recommended action to resolve the problem. Note that the `bpixmaptext` utility will call `edcmtext` when the `errno2` value is in the range reserved for the C run-time library (0xC0000000 through 0xCFFFFFFF). The `edcmtext` utility can also be called independent of `bpixmaptext`. IFigure 28-6 shows an example of calling `edcmtext`.

```
LUTZ:/u/lutz: >edcmtext C9330011

JrEdcWexpEnospace01: Attempt to allocate an internal buffer failed

Action: An internal call to malloc failed. Look for errno value set
for further information about the cause of this failure.

Source: edcowexp.c

LUTZ:/u/lutz: >
```

Figure 28-6 Example of calling `edcmtext`

Part of the work involved in making the `edcmtext` utility useful to the customer is the addition of `errno2` values into the C run-time library. The addition of these values makes it beneficial to change the default of `perror()` messages to display both the `errno` and `errno2` values.

The display of the `errno2` value in the messages is controlled by the value of the `_EDC_ADD_ERRNO2` variable. Thus, the `_EDC_ADD_ERRNO2` variable will be changed such that when it is unset, `errno2` will be added to `perror()` messages.

### fdlibm replacement

With z/OS V1R9, certain IEEE754 `fdlibm` math functions are replaced. The earlier versions of functions that are more closely aligned with the C99 standard are no longer available. Neither the `_IEEEV1_COMPATIBILITY` feature test macro nor the `_EDC_IEEEV1_COMPATIBILITY` environment variable can be used to affect these functions.

The earlier versions of functions with performance and accuracy enhancements are still available. To use earlier versions of the IEEE754 `fdlibm` math functions, use either of the following methods:

- ▶ When using the `FLOAT(IEEE)` compiler option, use the `_IEEEV1_COMPATIBILITY` feature test macro.
- ▶ When variable mode is in effect, use environment variable `_EDC_IEEEV1_COMPATIBILITY_ENV=ON`.

**Note:** To modify your source code to use the new performance and accuracy enhancements, use the information in Table 18 in *z/OS XL C/C++ Compiler and Run-Time Migration Guide for the Application Programmer*, GC09-4913.

## 28.4 HEAPPOOLS performance improvement

The `HEAPPOOLS` and `HEAPPOOLS64` run-time options is used to control an optional heap storage management algorithm known as *heap pools*. This algorithm is designed to improve performance of multi threaded C/C++ applications with high usage of `malloc()`, `calloc()`, `realloc()` and `free()`. When active, heap pools virtually eliminates contention for heap storage.

The problem in the past was the cache contention that occurs when two CPs try to update the same storage address at the same time. This contention may be unavoidable. But cache contention may also occur when two CPs update different storage addresses that are close together at the same time.

This new ALIGN option eliminates some of this type of contention by rearranging how the heap pool structures are laid out in storage. However, doing so uses more virtual storage. Figure 28-7 shows the syntax of the new HEAPPOOLS runtime options.

```
HEAPPOOLS64=(ALIGN | ON | OFF,.....
```

Figure 28-7 Syntax of the modified HEAPPOOLS option

### HEAPPOOLS64 options

Table 28-5 lists the descriptions of all possible values of the HEAPPOOLS parameter.

Table 28-5 New options for HEAPPOOLS

Value	Meaning
ON	Specifies that Language Environment uses the Heap Pool Manager to manage heap storage requests against the initial heap.
OFF	Specifies that Language Environment does not use the Heap Pools Manager.
ALIGN	Specifies that Language Environment structures the storage for cells in a heap pool so that a cell less than or equal to 248 bytes, or 240 bytes for 64-bit, does not cross a cache line. For cells larger than 248 bytes, two cells never share a cache line.

**Important:** Using the ALIGN suboption might cause an increase in the amount of heap storage used by an application. Examine the storage report and adjust storage tuning when first using the ALIGN suboption.

For AMODE 64 applications, heap pool cells are now always aligned on quadword boundaries. This will cause an increase in virtual storage usage from prior releases for cell sizes that are not a multiple of 16.

### Migration considerations

When an application linked with a CEEUOPT or CELQUOPT that specified ALIGN for the first suboption of the HEAPPOOLS or HEAPPOOLS64 runtime option is executed on lower releases, ALIGN is treated like ON. There is no toleration APAR associated with this enhancement.

As mentioned, for AMODE 64 applications, heap pool cells are now always aligned on quadword boundaries, which causes an increase in virtual storage usage from prior releases for cell sizes that are not a multiple of 16.

## 28.5 z/OS UNIX support for ceebltx utility

The ceebltx utility creates several files from the message source file. It creates an assembler source file, which can be assembled into an object (text) file and link-edited into a

module in an MVS load library. When the name of the module is placed in a message module table, the Language Environment message services can dynamically access the messages.

The ceebltx utility optionally creates secondary input files, which contain declarations for the condition tokens associated with each message in the message source file. When a program uses the secondary input file, the condition tokens can then be used to reference the messages from the message table. The :msgname. tag indicates the symbolic name of the condition token. The syntax for calling the ceebltx utility is shown in Figure 28-8. In prior releases of z/OS, the CEEBLDTX only shipped as a CLIST.

```
ceebltx

      [-C csect_name] [-I secondary_file_name]
      [-P] [-S] [-c class] [-d APOST | ' | QUOTE | "]
      [-l BAL | C | COBOL | FORTRAN | PLI] [-s id]
      in_file out_file
```

Figure 28-8 Syntax of the ceebltx utility

Where:

- |                               |  |
|-------------------------------|--|
| <b>in_file</b>                | The name of the file containing the message source.  |
| <b>out_file</b>               | The name of the resulting assembler source file containing the messages, inserts, and others items, suitable for input into the High Level Assembler. An extension of .s is assumed if none is present.  |
| <b>-C csect_name</b>          | This option is used to explicitly specify the CSECT name. An upper case version of the CSECT name will be used. By default, the CSECT name is the output file base name.   |
| <b>-I secondary_file_name</b> | The -I (uppercase i) option provides the name of the secondary input file generated for the language specified with the -l (lower case L) option.<br><br>If no suffix is present in the secondary_file_name specified, the extension will be .h for C, .fortran for FORTRAN, and .copy for all others.           |
| <b>-P</b>                     | This option is used to save previous prologs, if files being generated already exist in the directory and contain prologs. By default, previous prologs are not reused.  |
| <b>-S</b>                     | This option is used to indicate sequence numbers should be generated in the files produced. By default, no sequence numbers are generated.   |
| <b>-c class</b>               | The class option is used to specify the default value for :msgclass in cases where the tag is not coded.   |
| <b>-d</b>                     | APOST   '   QUOTE   " - This option is used to specify which COBOL delimiter to use and is used in combination with the -l (lower case L) COBOL option. By default, APOST is used as the delimiter.  |
| <b>-l</b>                     | BAL   C   COBOL   FORTRAN   PL1 - The -l (lower case L) option is used to specify the language to be used in generating a secondary input file and is used in combination with the I secondary_file_name option. The file will contain declarations for the condition tokens associated with each message in the |

message source file. The language is accepted in lower case and upper case. C370 is also supported.

**-s id**

The `id` option is used to specify the default value for `:msgsubid.` in cases where the tag is not coded.

**Restriction:** The `ceebldtx` utility only works with z/OS UNIX files. MVS data sets are not applicable.

## 28.6 CLER run-time option change support

The CICS transaction CLER run-time option allows you to display all the current Language Environment run-time options for a region, and to also have the capability to modify a subset of these options. The following run-time options can be modified with the CLER transaction:

- ▶ ALL31(ONIOFF)
- ▶ CBLPSHPOP(ONIOFF)
- ▶ CHECK(ONIOFF)
- ▶ INFOMSGFILTER(ONIOFF)
- ▶ RPTOPTS(ONIOFF)
- ▶ RPTSTG(ONIOFF)
- ▶ TERMTHDACT(QUIETIMSGITRACEIDUMPIUAONLYIUATRACEI UADUMPIUAIMM)
- ▶ TRAP(ONIOFF)

Setting `RPTOPTS(ON)` or `RPTSTG(ON)` in a production environment can significantly degrade performance. Also, if `ALL31(OFF)` is set in a production environment, the stack location will be set to below the 16 MB line, which could cause the CICS region to abend due to lack of storage. The LAST WHERE SET column of the Language Environment run-time options report will contain CICS CLER Trans for those options that were set by CLER.

**Important:** CICS TS 3.1 and higher supports XPLINK programs in a CICS environment. The CLER transaction does not affect the run-time options for these programs.

## 28.7 New and modified callable services

z/OS V1R8 provides new callable services for Language Environment applications. These callable services can be invoked from applications generated with the following IBM compiler products:

- ▶ IBM z/OS XL C/C++
- ▶ C/C++ for MVS/ESA™
- ▶ IBM SAA® AD/Cycle® C/370™
- ▶ Enterprise COBOL for z/OS
- ▶ Enterprise PL/I for z/OS
- ▶ IBM COBOL for OS/390 and VM
- ▶ IBM COBOL for MVS and VM
- ▶ IBM PL/I for MVS and VM

You can also invoke the Language Environment callable services from assembler programs that use the `CEEENTRY` and associated macros.

## CEE3MC2 callable service

The CEE3MC2 callable service is similar to the CEE3MCS callable service. The new service provides the current currency symbol and the international currency symbol. The usage is shown in Figure 28-9.

```
CEE3MC2 ( country_code , currency_symbol , international_currency_symbol , fc )
```

Figure 28-9 Format of the CEE3MC2 callable service

The CEE3MC2 callable service has the following call and return parameter:

**country\_code (input)**

This is a 2-character fixed-length string representing one of the country codes found in Table 28 in the *z/OS Language Environment Programming Reference, SA22-7562*. `country_code` is not case-sensitive. If no value is specified, then the default country code, as set by the COUNTRY run-time option or the CEE3CTY callable service, is used.

**currency\_symbol (output)**

This is a 4-character fixed-length string returned to the calling routine. It contains the default currency symbol for the country specified. The currency symbol is left-justified and padded on the right with blanks, if necessary.

**international\_currency\_symbol (output)**

This is a 3-character alphabetic fixed-length string returned to the calling routine. It contains the international currency symbol for the country specified.

**fc (output)**

This is a feedback code, optional in some languages, that indicates the result of this service.

The following Feedback Codes can result from this service:

**CEE000** The service completed successfully.

**CEE3C2** The country code `country_code` was invalid for CEE3MC2. The default currency symbol `currency_symbol` was returned. No international currency symbol was returned.

**Note:** The default currency symbol for the US is \$. The international currency symbol for the US is USD.

## 28.8 CEE3DLY and CEEDLYM callable services

The CEE5DLY functionality is currently a LE/VSE callable service. CEE5DLY enables a LE-conforming program to suspend execution for a specified number of seconds. There is no equivalent functionality in Language Environment on z/OS, causing portability concerns for VSE applications using CEE5DLY. The existing code currently uses ILBOWAT0 (from a pre-LE COBOL run-time library) routine, which requires an AMODE24 parameter below the 16 M line, leading to virtual constraint problems. Sites are currently required to develop and maintain in-house assembler routines. It would be easier to have a similar callable service on

z/OS that would facilitate portability for VSE applications and eliminate the additional tasks of creating and maintaining assembler routines. For more flexibility, a separate callable service, CEE3DLYM, is created for times lower than a second.

Both callable services put the active enclave to sleep for the given time. The CEE3DLY callable service expects the time in seconds as the first parameter; see Figure 28-10.

```
CEE3DLY ( seconds, fc)
```

Figure 28-10 Format of the CEE3DLY callable service

The CEE3DLY callable service expects the time in milliseconds as the first parameter; see Figure 28-11.

```
CEEDLYM ( milliseconds, fc)
```

Figure 28-11 Format of the CEE3DLYM callable service

## 28.9 AMODE 64 CELQPIPI service vector

Language Environment preinitialization (PreInit) is commonly used to enhance performance for repeated invocations of an application or for a complex application where there are many repetitive requests and where fast response is required. For instance, if an assembler routine invokes either a number of Language Environment-conforming HLL routines or the same HLL routine a number of times, the creation and termination of that HLL environment multiple times is needlessly inefficient. A more efficient method is to create the HLL environment only once for use by all invocations of the routine.

PreInit lets an application initialize an HLL environment once, perform multiple executions using that environment, and then explicitly terminate the environment. Because the environment is initialized only once (even if you perform multiple executions), you free up system resources and allow for faster responses to your requests.

In the 64-bit environment, CELQPIPI provides the interface for preinitialized routines. Using CELQPIPI, you can initialize an environment, invoke applications, terminate an environment, and add an entry to the PreInit table. (The PreInit table contains the names and entry point addresses of routines that can be executed in the preinitialized environment.)

In previous releases, when running in AMODE 31, the PREINIT LE facility could replace the IBM-supplied LOAD, DELETE, GETSTOR, FREESTOR, EXCEPTRTN and MSGRTN routines using the CELQPIPI interface.

### AMODE 64 support

z/OS V1R9 provides support for running in AMODE 64, and that exploits the PREINIT LE facility to replace the IBM-supplied LOAD and DELETE using the CELQPIPI interface if the loaded modules are obtained from a non-hierarchical file system (HFS).

The AMODE 64 LOAD and DELETE routines can perform additional useful processing each time Language Environment LOADs or DELETEs a non-HFS module (for example, logging, caching, substitution, statistics, accounting, diagnostics, and so on).

This support is invoked using one assembler PreInit driver program that calls macro CELQPIPI to start the PreInit interface:

```
Call CELQPIPI(init_xxx, ,service_rtns, )
```

The previously-reserved third parameter for CELQPIPI can now optionally point to a service routine vector (similar to AMODE 31 CEEPIPI calls).

To specify replacement LOAD and DELETE routines, the third parameter of the CELQPIPI INIT\_MAIN or INIT\_SUB call must point to an AMODE 64 Service routine vector.

The layout of this service routine vector is shown in the Language Environment Programming Guide for 64-bit Virtual Addressing Mode. In simplified terms, the layout is:

- ▶ 4 bytes of zeros
- ▶ Count of doublewords that follow in the vector
- ▶ Userword (passed through to the LOAD and DELETE routines)
- ▶ Pointer to AMODE 64 Function Descriptor for the LOAD replacement routine (or 0 = no replacement)
- ▶ Pointer to AMODE 64 Function Descriptor for the DELETE replacement routine (or 0 = no replacement)
- ▶ Five more doublewords of zeros (reserved) – optional, but must be present if the count is 9

The LOAD and DELETE replacement routines get control in AMODE64 with the usual register and parameter passing conventions. The stack to be used is usually the normal Language Environment stack, but in some cases may be a special fixed-sized stack with room for a 4096-byte DSA (dynamic storage area).

The LOAD and DELETE routines need to pass back output and return to Language Environment. Either both or neither of the LOAD and DELETE routines must be replaced (it is not possible to replace just LOAD or just DELETE).

In prior releases, AMODE 64 PreInit applications may have gotten away with non-zero contents in the reserved third parameter of CELQPIPI INIT\_MAIN or INIT\_SUB calls. Now, this third parameter must be 0 or point to a valid service routine vector (similar to AMODE 31).

## 28.10 AMODE 64 CEETBCK and CEEHGOTO

CEETBCK and CEEHGOTO services are available for AMODE 31 applications, and not to AMODE 64 applications.

- ▶ This release changed the `_far_jump()` function to include the CEEHGOTO services in AMODE 64 C runtime library. `_far_jump()` is already supported for XPLINK.

The function `__far_jump()` can be used in a signal catcher, exception handler, or debugger to return control to the application.

- ▶ There is a new AMODE 64 only, C runtime library interface: `_le_traceback()` that is equivalent to the CEETBCK callable services.

The function `__le_traceback()` can be used to generate a traceback instead of calling `ctrace()` or `cdump()` when the user wants a different format or different location for the traceback.



## 28.10.1 `__far_jump()` function

The `__far_jump()` interface performs a function similar to `longjmp()`. However, it does not require a `setjmp()` to be performed previously. The information required to perform this nonlocal goto is provided by the user in the `__jumpinfo` structure. This information includes registers and signal mask.

The target address of the jump is not supplied separately. It is supplied as two of the register values in the GPR set in the `__jumpinfo` structure: Register 4 for the target DSA address, and Register 7 for the target code address.

The format is:

```
#include <edcwccwi.h>
void __far_jump (struct __jumpinfo * JumpInfo);
```

## 28.10.2 `__le_traceback()` function

The `__le_traceback()` function assists in tracing the call chain. It identifies the language, program unit, entry point, current location, caller's DSA, and other information from the address of a DSA for a program unit. This is essential for creating meaningful traceback messages.

The format is:

```
#include <__le_api.h>
void __le_traceback(int cmd, void* cmd_parms, _FEEDBACK *fc);
```

The `__le_traceback()` function uses a single structure and the arguments are:

- |                        |   |
|------------------------|---|
| <b>int cmd</b>         | The following <code>__le_traceback()</code> command is used: <code>__TRACEBACK_FIELDS</code> information that can be used to create a traceback message is returned in individual fields.   |
| <b>void* cmd_parms</b> | A pointer to a structure that contains additional command-specific parameters. For the command <code>__TRACEBACK_FIELDS</code> , this parameter must point to a <code>__tf_parms_t</code> . |
| <b>_FEEDBACK* fc</b>   | A 16-byte feedback code is returned in this parameter.  |

## 28.11 XPLINK enhancements

Extra performance linkage (XPLINK) is a call linkage between programs that has the potential for a significant performance increase when used in an environment of frequent calls between small functions or subprograms.

The objective of XPLINK is to significantly speed up the linkage for C and C++ routines by using a downward-growing stack and by passing parameters in registers. It includes support for reentrant and non-reentrant code, for calls to functions in DLLs, and compatibility with old code.

With XPLINK, the linkage and parameter passing mechanisms for C and C++ are identical. If you link to a C function from a C++ program, you should still specify `extern C` to avoid name mangling.

The primary objective of XPLINK is to make subroutine calls as fast and efficient as possible by removing all nonessential instructions from the main path.

XPLINK has been improved to support new environments. z/OS V1R9 supports application running in AMODE 31 using the XPLINK and with Library Routine Retention (LRR) in an IMS environment.

IMS applications can now take advantage of XPLINK (with or without LRR) if required. AMODE 31 XPLINK applications can take advantage of LRR (without IMS), if required.

The support is invoked by a non-LE-enabled assembler program, as shown in Figure 28-12.

```
CEELRR ACTION=INIT,XPLINK=YES
```

*Figure 28-12 CEELRR callable service*

## Metal option of XL C compiler

This appendix provides different cases of the Metal option of the XL C compiler that have been tested.

- ▶ JCL procedure for CATALG
- ▶ Example of using CATALG

## A.1 JCL procedure METACALG

Example A-1 shows a JCL procedure called METACALG that is used to run the Metal option.

*Example: A-1 C-compile, assemble, link-edit,go*

```
//METACALG PROC INFILE=, < INPUT ... REQUIRED
// ASMFIL=, < OUTPUT C ... REQUIRED
// CRUN=, < COMPILER RUNTIME OPTIONS
// CPARM=, < COMPILER OPTIONS
// CPARM2=, < COMPILER OPTIONS
// CPARM3=, < COMPILER OPTIONS
// LIBPRFX='CEE', < PREFIX FOR LIBRARY DSN
// LNGPRFX='CBC', < PREFIX FOR LANGUAGE DSN
// CLANG='EDCMSGE', < NOT USED IN THIS RELEASE. KEPT FOR COMPATIBILITY
// DCB80='(RECFM=FB,LRECL=80,BLKSIZE=3200)', <DCB FOR LRECL 80
// DCB3200='(RECFM=FB,LRECL=3200,BLKSIZE=12800)', <DCB FOR LRECL 3200
// TUNIT='SYSALLDA', < UNIT FOR TEMPORARY FILES
// TSPACE='(32000,(30,30))' < SIZE FOR TEMPORARY FILES
//*
/*****
/*** METACALG
/**
/*****
/*
/* THIS PROCEDURE
/*
/* 1-COMPILES A C PROGRAM WITH z/OS XL C/C++ AND THE Metal OPTION
/* 2-RUNS THE HIGH LEVEL ASSEMBLER,
/* 3-LINK-EDITS THE NEWLY ASSEMBLED PROGRAM
/* 4-RUNS THE PROGRAM WHEN THE LINK-EDIT IS SUCCESSFULLY ACCOMPLISHED.
/*
/*
/*-----
/* C COMPILE STEP:
/*-----
/*C EXEC PGM=CCNDRVR,REGION=144M,
// PARM=('&CRUN/Metal &CPARM &CPARM2 &CPARM3')
//STEPLIB DD DSNAME=&LIBPRFX..SCEERUN2,DISP=SHR
// DD DSNAME=&LNGPRFX..SCNCMP,DISP=SHR
// DD DSNAME=&LIBPRFX..SCEERUN,DISP=SHR
//SYSMSGSD DUMMY,DSN=&LNGPRFX..SCBC3MSG(&CLANG),DISP=SHR
//SYSIN DD DSNAME=&INFILE,DISP=SHR
//SYSLIB DD DSNAME=&LIBPRFX..SCEEH.H,DISP=SHR
// DD DSNAME=&LIBPRFX..SCEEH.SYS.H,DISP=SHR
//SYSLIN DD DSNAME=&ASMFIL,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSCPRT DD SYSOUT=*
//SYSUT1 DD UNIT=&TUNIT.,SPACE=&TSPACE.,DCB=&DCB80
//SYSUT5 DD UNIT=&TUNIT.,SPACE=&TSPACE.,DCB=&DCB3200
//SYSUT6 DD UNIT=&TUNIT.,SPACE=&TSPACE.,DCB=&DCB3200
//SYSUT7 DD UNIT=&TUNIT.,SPACE=&TSPACE.,DCB=&DCB3200
//SYSUT8 DD UNIT=&TUNIT.,SPACE=&TSPACE.,DCB=&DCB3200
//SYSUT9 DD UNIT=&TUNIT.,SPACE=&TSPACE.,
```

```

//          DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10 DD SYSOUT=*
//SYSUT14 DD UNIT=&TUNIT.,SPACE=&TSPACE.,
//          DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT16 DD UNIT=&TUNIT.,SPACE=&TSPACE.,
//          DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT17 DD UNIT=&TUNIT.,SPACE=&TSPACE.,
//          DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//*
//*-----
//* ASSEMBLE STEP:
//*-----
//A          EXEC PGM=ASMA90,PARM=(OBJECT,NODECK),COND=(8,LT,C)
//SYSLIB    DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1    DD DSN=&&SYSUT1,SPACE=(4096,(120,120),,ROUND),UNIT=VIO,
//          DCB=BUFNO=1
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD SYSOUT=B
//SYSLIN    DD DSN=&&OBJ,SPACE=(3040,(40,40),,ROUND),UNIT=VIO,
//          DISP=(MOD,PASS),
//          DCB=(BLKSIZE=3040,LRECL=80,RECFM=FBS,BUFNO=1)
//SYSIN     DD DSN=&ASMFIL,DISP=SHR
//*
//*-----
//* LINK-EDIT STEP:
//*-----
//L          EXEC PGM=HEWL,PARM='MAP,LET,LIST,NCAL',COND=(8,LT,A)
//STEPLIB  DD DSN=&LIBPRFX..SCEERUN2,DISP=SHR
//          DD DSN=&LNGPRFX..SCCNCMP,DISP=SHR
//          DD DSN=&LIBPRFX..SCEERUN,DISP=SHR
//SYSLIN    DD DSN=&&OBJ,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//SYSLMOD  DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(1,1,1)),
//          DSN=&&GOSET(GO)
//SYSUT1    DD DSN=&&SYSUT1,SPACE=(1024,(120,120),,ROUND),UNIT=VIO,
//          DCB=BUFNO=1
//SYSPRINT DD SYSOUT=*
//*
//*-----
//* GO STEP:
//*-----
//G          EXEC PGM=*.L.SYSLMOD,COND=((8,LT,C),(8,LT,L))

```

## METACALG example

Example A-2 illustrates the JCL for using METACALG.

*Example: A-2 JCL for using METACALG*

```

//LAFITTEG JOB 'LAFITTE',MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*JOBPARM SYSAFF=SC63
//*****
//*****
//Meta1 EXEC PROC=METACALG,
//          INFILE='LAFITTE.ESSAIS.C(STRCOPY)',
//          ASMFIL='LAFITTE.ESSAIS.ASM(STRCOPY)',

```

```
//      CPARAM='LSEARCH(''''LAFITTE.HDRS.+''''')',  
//      PARM.A='NOLIBMAC',  
//      PARM.L='MAP,LET,LIST,INFO,MODMAP(LOAD)'  
//
```

---

**Note:** In procedure METACALG, there is a variable &ASMFILE for a dataset (and its member) in which the assemble code generated by the C-compiler is set aside. It simply permits you to examine the code generated but has no value by itself and can easily be changed into a temporary file passed between the C-compile and assemble steps.

## **System REXX for z/OS**

This appendix contains a System REXX exec.

- ▶ WHOIAM REXX exec

## B.1 REXX exec WHOIAM

Example B-1 is a System REXX exec that illustrates how to determine, from within the exec itself, which kind of environment it is running into.

*Example: B-1 WHOIAM*

```
/* */
/* This is an example of the logic to develop */
/* in order to figure, from within an exec, */
/* whether it has been invoked */
/* from classical TSO environment (IKJJEFT01)*/
/* from Console thru F AXR command */
/* from AXREXX API interface */
/* */
/* Then Say it to the world */
/* */
/* J.-L. Lafitte*/
/* 05/22/2007 */
/* */
/* Do not go too far, without asking obvious question */

If AXREQTOKEN ^= 'AXREQTOKEN' Then Do;
  Say 'Invoked from TSO/E classical environment'
  Exit 0;
End;

/* Now we should be in the SYSTEM REXX envrt */

MyName = 'WHOIAM';
MyCmd = 'F AXR,SYSREXX ST,D';

Result = AXRCMD(MyCmd, OutputVar. ,10);

If Result = 0 Then Do;

  CallerName = ' '

  Do LineNum = 1 to OutputVar.0 While(CallerName=' ');
  Parse var OutputVar.LineNum 'EXEC=' MyName Rest.1 ;
  Parse var Rest.1 'CJBN=' CallerName Rest.2 ;
  Parse var Rest.2 'TSO=' YesorNo Rest.3 ;

  If CallerName ^= ' ' Then Do;

    line2 =LineNum+1
    Parse var OutputVar.line2 'REQTOKEN=' MyToken Rest.9 ;

    If MyToken = AXRREQTOKEN Then Do ;

      If CallerName = 'AXR' Then Do;
        If YesorNo ^= 'Y' Then Do;
          Say 'There is a problem, a SYSREXX exec invoked '
          Say 'from Console, cannot be but TSO=YES'
          Exit 20;
        End;
      End;
    End;
  End;
End;
```



```

        End;
        Else Do;
        Say 'Invoked from Console with TSO=YES';
        Exit 0;
        End;
    End;
    Else Do;
    End;
    If YesorNo = 'Y' Then Do;
        Say 'Invoked thru API by ' CallerName ' with TSO=YES';
        Exit 0;
    End;
    Else Do;
        Say 'Invoked thru API by ' CallerName ' with TSO=NO';
        Exit 0;
    End;
End;
End;
End;
If CallerName = ' ' Then Do;
    MyRetcode = 24;
    Say 'Something wrong, couldnot find self'
End;
Else Do;          /* CallerName not equ ' ' */
    MyRetcode = 28;
    say 'Something went wrong, should never be here'
End;
End;          /* end of cmd result equ 0 */

Else Do;          /* cmd result not equ 0 */
    MyRetcode = 32;
    Say ' Command ' MyCmd ' did not go right';
End;

Exit MyRetcode;          /* should come here ONLY */
                        /* when somethg went wrong */

```

Example B-2 illustrates a more system-oriented exec which finds its ASTE address from a given JOBNAME.

*Example: B-2 Locate the ASTE of 'JOBNAME'*

---

```

/*      From a JobName                                     */
/*      Get Address of Primary Adress Space ASTE         */
/*      Get Address of Primary Adress Space ASTE         */
/*      Assign it to var OutAste@ var                     */
/*      so it can be retrieved by an ASM pgm             */
/*      if such is the caller                             */
/*                                                         */
NUMERIC DIGITS 25
ARG InJobname
MyCmd = 'D JOBS,' || Strip(InJobname);
Result = AXRCMD(MyCmd, OutputVar. ,10);
IF Result = 0 THEN
DO;
    OutAste@ = ' '
    DO LineNum = 1 TO OutputVar.0 WHILE(OutASTE@=' ');

```

```
    PARSE var OutputVar.LineNum 'ASTE=' OutAste@
END;
IF OutAste@ = ' ' THEN
    DO;
        MyRetcode = 8;
        OutAste@ = 0;
    END;
ELSE /* OutAste@ not equ ' ' */
    MyRetcode = 0;
    say 'ASTE <--' OutAste@
END;
ELSE /* result not equ 0 */
    DO;
        MyRetcode = 12;
        OutAste@ = 0;
    END;
EXIT MyRetcode;
```

---



## **z/OS Communications Server**

This appendix provides information related to z/OS Communications Server with z/OS V1R9.

- ▶ IPSEC policy configuration for SC70
- ▶ IPSEC policy configuration for SC65
- ▶ SC65 pbr configuration files
- ▶ SC70 pbr configuration files
- ▶ SC65 netstat -A command
- ▶ SC70 netstat -A command
- ▶ pasearch -R command

## C.1 IPSEC policy configuration for SC70

*Example: C-1 SC70 IPSEC policy configuration*

---

```
##
## IPsec Policy Agent Configuration file for:
##   Image: SC70
##   Stack: TCPIP
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Fri May 04 10:08:14 EDT 2007
##
## Copyright = None
##

## NOTE -- Generated IpGenericFilterAction Permit~LogNo
IpGenericFilterAction      Permit~LogNo
{
  IpFilterAction           Permit
  IpFilterLogging          No
}

IpGenericFilterAction      IpSec~LogNo
{
  IpFilterAction           IpSec
  IpFilterLogging          No
}

KeyExchangeOffer          KE0~1
{
  HowToEncrypt             3DES
  HowToAuthMsgs            SHA1
  HowToAuthPeers           PresharedKey
  DHGroup                  Group1
  RefreshLifetimeProposed  480
  RefreshLifetimeAccepted  240 1440
  RefreshLifesizeProposed  None
  RefreshLifesizeAccepted  None
}

IpDataOffer               IPsec__Gold~R
{
  HowToEncap               Transport
  HowToEncrypt             3DES
  HowToAuth                ESP Hmac_Sha
  RefreshLifetimeProposed  240
  RefreshLifetimeAccepted  120 480
  RefreshLifesizeProposed  None
  RefreshLifesizeAccepted  None
}

IpDataOffer               IPsec__Gold~R~2
{
  HowToEncap               Transport
```

```

HowToEncrypt          AES
  HowToAuth           ESP Hmac_Sha
  RefreshLifetimeProposed 240
  RefreshLifetimeAccepted 120 480
  RefreshLifesizeProposed None
  RefreshLifesizeAccepted None
}

## NOTE -- Generated IpService IKE~Gen
IpService             IKE~Gen
{
  Protocol            UDP
  SourcePortRange     500
  DestinationPortRange 500
  Direction           BiDirectional
  Routing             Local
}

IpService             All_other_traffic
{
  Protocol            All
  Direction           BiDirectional
  Routing             Local
}

IpService             All_other_traffic~3
{
  Protocol            All
  Direction           BiDirectional
  Routing             Either
}

IpDynVpnAction       IPsec__Gold
{
  Initiation          Either
  VpnLife             1440
  Pfs                 None
  IpDataOfferRef     IPsec__Gold~R
  IpDataOfferRef     IPsec__Gold~R~2
}
##
## Connectivity Rule 2 combines the following items:
## Local data endpoint 2~ADR~1
## Remote data endpoint 2~ADR~2
## Topology            HH
## Requirement Map     VPN7065
## All_other_traffic  => IPsec__Gold

IpAddr               2~ADR~1
{
  Addr                9.12.4.202
}

IpAddr               2~ADR~2
{

```

```

    Addr                9.12.4.48
  }

LocalSecurityEndpoint  2~LSE~4
{
  Identity              IpAddr 9.12.4.202
  LocationRef           2~ADR~1
}

RemoteSecurityEndpoint 2~RSE~3
{
  Identity              IpAddr 9.12.4.48
  LocationRef           2~ADR~2
}

KeyExchangeRule       2~5
{
  LocalSecurityEndpointRef 2~LSE~4
  RemoteSecurityEndpointRef 2~RSE~3
  KeyExchangeActionRef     2
  SharedKey                 Ebcdic iked
}

KeyExchangeAction     2
{
  HowToInitiate         Main
  HowToRespond          Either
  KeyExchangeOfferRef   KE0~1
  AllowNat              No
}

## NOTE -- Generated IpFilterRule 2~6
IpFilterRule          2~6
{
  IpSourceAddrRef       2~ADR~1
  IpDestAddrRef         2~ADR~2
  IpServiceRef          IKE~Gen
  IpGenericFilterActionRef Permit~LogNo
}

IpFilterRule          2~7
{
  IpSourceAddrRef       2~ADR~1
  IpDestAddrRef         2~ADR~2
  IpServiceRef          All_other_traffic
  IpGenericFilterActionRef IpSec~LogNo
  IpDynVpnActionRef     IPSec__Gold
}

##
## Connectivity Rule 1 combines the following items:
## Local data endpoint    All4
## Remote data endpoint   All4
## Topology               None (Permit/Deny only)
## Requirement Map        Permit_All
## All_other_traffic      => Permit

```

```

IpFilterRule                1~1
{
  IpSourceAddr              A114
  IpDestAddr                A114
  IpServiceRef              All_other_traffic~3
  IpGenericFilterActionRef  Permit~LogNo
}

KeyExchangePolicy
{
  AllowNat                  No
  KeyExchangeRuleRef       2~5
}

IpFilterPolicy
{
  PreDecap                  OFF
  FilterLogging             ON
  IpFilterLogImplicit       No
  AllowOnDemand             Yes
  IpFilterRuleRef          2~6
  IpFilterRuleRef          2~7
  IpFilterRuleRef          1~1
}

```

---

## C.2 IPSEC policy configuration for SC65

*Example: C-2 SC65 IPSEC policy configuration*

---

```

##
## IPsec Policy Agent Configuration file for:
##   Image: SC65
##   Stack: TCPIP
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 8
## Date Created: Fri May 04 10:08:44 EDT 2007
##
## Copyright = None
##

## NOTE -- Generated IpGenericFilterAction Permit~LogNo
IpGenericFilterAction      Permit~LogNo
{
  IpFilterAction           Permit
  IpFilterLogging          No
}

IpGenericFilterAction      IpSec~LogNo
{
  IpFilterAction           IpSec
}

```

```

    IpFilterLogging          No
  }
  KeyExchangeOffer         KE0~1
  {
    HowToEncrypt            3DES
    HowToAuthMsgs          SHA1
    HowToAuthPeers         PresharedKey
    DHGroup                 Group1
    RefreshLifetimeProposed 480
    RefreshLifetimeAccepted 240 1440
    RefreshLifesizeProposed None
    RefreshLifesizeAccepted None
  }

  IpDataOffer              IPsec__Gold~R
  {
    HowToEncap             Transport
    HowToEncrypt           3DES
    HowToAuth              ESP Hmac_Sha
    RefreshLifetimeProposed 240
    RefreshLifetimeAccepted 120 480
    RefreshLifesizeProposed None
    RefreshLifesizeAccepted None
  }

  IpDataOffer              IPsec__Gold~R~2
  {
    HowToEncap             Transport
    HowToEncrypt           AES
    HowToAuth              ESP Hmac_Sha
    RefreshLifetimeProposed 240
    RefreshLifetimeAccepted 120 480
    RefreshLifesizeProposed None
    RefreshLifesizeAccepted None
  }

  ## NOTE -- Generated IpService IKE~Gen
  IpService                 IKE~Gen
  {
    Protocol               UDP
    SourcePortRange        500
    DestinationPortRange   500
    Direction              BiDirectional
    Routing                 Local
  }

  IpService                 All_other_traffic
  {
    Protocol               All
    Direction              BiDirectional
    Routing                 Local
  }

  IpService                 All_other_traffic~3
  {
    Protocol               All

```



```

    Direction                BiDirectional
    Routing                   Either
}

IpDynVpnAction              IPsec__Gold
{
    Initiation               Either
    VpnLife                  1440
    Pfs                      None
    IpDataOfferRef           IPsec__Gold~R
    IpDataOfferRef           IPsec__Gold~R~2
}
##
## Connectivity Rule 2 combines the following items:
## Local data endpoint      2~ADR~1
## Remote data endpoint     2~ADR~2
## Topology                 HH
## Requirement Map          VPN6570
## All_other_traffic        => IPsec__Gold

IpAddr                      2~ADR~1
{
    Addr                     9.12.4.48
}

IpAddr                      2~ADR~2
{
    Addr                     9.12.4.202
}

LocalSecurityEndpoint       2~LSE~4
{
    Identity                 IpAddr 9.12.4.48
    LocationRef              2~ADR~1
}

RemoteSecurityEndpoint      2~RSE~3
{
    Identity                 IpAddr 9.12.4.202
    LocationRef              2~ADR~2
}

KeyExchangeRule             2~5
{
    LocalSecurityEndpointRef 2~LSE~4
    RemoteSecurityEndpointRef 2~RSE~3
    KeyExchangeActionRef     2
    SharedKey                 EbcDic iKed
}

KeyExchangeAction           2
{
    HowToInitiate            Main
    HowToRespond             Either
    KeyExchangeOfferRef      KE0~1
}

```

```

    AllowNat                No
}

## NOTE -- Generated IpFilterRule 2~6
IpFilterRule                2~6
{
    IpSourceAddrRef        2~ADR~1
    IpDestAddrRef          2~ADR~2
    IpServiceRef            IKE~Gen
    IpGenericFilterActionRef  Permit~LogNo
}

IpFilterRule                2~7
{
    IpSourceAddrRef        2~ADR~1
    IpDestAddrRef          2~ADR~2
    IpServiceRef            All_other_traffic
    IpGenericFilterActionRef  IpSec~LogNo
    IpDynVpnActionRef       IPSec__Gold
}
##
## Connectivity Rule 0 combines the following items:
## Local data endpoint      A114
## Remote data endpoint     A114
## Topology                  None (Permit/Deny only)
## Requirement Map          Permit_All
## All_other_traffic        => Permit

IpFilterRule                0~1
{
    IpSourceAddr            A114
    IpDestAddr              A114
    IpServiceRef            All_other_traffic~3
    IpGenericFilterActionRef  Permit~LogNo
}

KeyExchangePolicy
{
    AllowNat                No
    KeyExchangeRuleRef      2~5
}

IpFilterPolicy
{
    PreDecap                OFF
    FilterLogging            ON
    IpFilterLogImplicit      No
    AllowOnDemand            Yes
    IpFilterRuleRef          2~6
    IpFilterRuleRef          2~7
    IpFilterRuleRef          0~1
}

```

---

## C.3 SC65 pbr configuration files

*Example: C-3 SC65 pbr configuration files*

---

```
##
## PBR Policy Agent Configuration file for:
##   Image: SC65
##   Stack: TCPIP
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 9
## Backing Store = C:\Program Files\IBM\zCSConfigAssist\V1R8\files\demo
## FTP History:
##
RoutingRule T070
{
    IpSourceAddr 9.12.4.48
    IpDestAddr 9.12.4.202
    TrafficDescriptorGroupRef T070~
    Priority 500000
    RoutingActionRef T070
}

## Action for RoutingRule:T070
RoutingAction T070
{
    UseMainRouteTable Yes
    RouteTableRef 70HS
}

RouteTable 70HS
{
    Route 9.12.4.202 10.1.101.70 IQDIOLNK0A016541 MTU 1500 NoRep1
}

TrafficDescriptorGroup T070~
{
    TrafficDescriptor
    {
        Protocol TCP
        SourcePortRange 0
        DestinationPortRange 0
        Jobname RODOLFI*
    }
}
```

---

## C.4 SC70 pbr configuration files

*Example: C-4 SC70 pbr configuration files*

---

```
##
## PBR Policy Agent Configuration file for:
```

```

## Image: SC70
## Stack: TCPIP
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 9
## Backing Store = C:\Program Files\IBM\zCSConfigAssist\V1R8\files\demo
## FTP History:
##
RoutingRule T065
{
    IpSourceAddr 9.12.4.202
    IpDestAddr 9.12.4.48
    TrafficDescriptorGroupRef T065~
    Priority 500000
    RoutingActionRef T065
}

## Action for RoutingRule:T065
RoutingAction T065
{
    UseMainRouteTable Yes
    RouteTableRef 65HS
}

RouteTable 65HS
{
    Route 9.12.4.48 10.1.101.65 IQDIOLNK0A016546 MTU 1500 NoRep1
}

TrafficDescriptorGroup T065~
{
    TrafficDescriptor
    {
        Protocol TCP
        SourcePortRange 0
        DestinationPortRange 0
        Jobname RODOLFI*
    }
}

```

---

## C.5 SC65 netstat -A command

*Example: C-5 SC65 netstat -A command*

```

RODOLFI @ SC65:/SC65/tmp>netstat -A -P 1953
MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIP          17:56:58
Client Name: RODOLFI3          Client Id: 000000C6
Local Socket: 9.12.4.48..1953  Foreign Socket: 0.0.0.0..0
  Last Touched:      15:16:14      State:      Listen
  BytesIn:           0000000000     BytesOut:   0000000000
  SegmentsIn:       0000000000     SegmentsOut: 0000000000
  RcvNxt:           0000000000     SndNxt:    0000000000

```

ClientRcvNxt:	000000000	ClientSndNxt:	000000000
InitRcvSeqNum:	000000000	InitSndSeqNum:	000000000
CongestionWindow:	000000000	SlowStartThreshold:	000000000
IncomingWindowNum:	000000000	OutgoingWindowNum:	000000000
SndWl1:	000000000	SndWl2:	000000000
SndWnd:	000000000	MaxSndWnd:	000000000
SndUna:	000000000	rtt_seq:	000000000
MaximumSegmentSize:	0000000536	DSField:	00
Round-trip information:			
Smooth trip time:	0.000	SmoothTripVariance:	1500.000
ReXmt:	000000000	ReXmtCount:	000000000
DupACKs:	000000000	RcvWnd:	0000032768
SockOpt:	00	TcpTimer:	00
TcpSig:	00	TcpSel:	00
TcpDet:	C0	TcpPol:	00
QOSPolicyRuleName:		SendBufferSize:	0000016384
RoutingPolicy:	No	ConnectionsDropped:	000000000
ReceiveBufferSize:	0000016384	MaximumBacklog:	0000000010
ConnectionsIn:	0000000001	SEF:	100
CurrentBacklog:	0000000000		
CurrentConnections:	0000000001		
Quiesced:	No		

----

Client Name: RODOLFI3	Client Id: 000000CC
Local Socket: 9.12.4.48..1953	Foreign Socket: 9.12.4.202..1027
Last Touched: 21:56:58	State: Estabsh
BytesIn: 2513414894	BytesOut: 000000000
SegmentsIn: 0196800828	SegmentsOut: 0068522197
RcvNxt: 0265183782	SndNxt: 2045797254
ClientRcvNxt: 0265183782	ClientSndNxt: 2045797254
InitRcvSeqNum: 2046736183	InitSndSeqNum: 2045797253
CongestionWindow: 0000005644	SlowStartThreshold: 0000065535
IncomingWindowNum: 0265216550	OutgoingWindowNum: 2045830022
SndWl1: 0265182383	SndWl2: 2045797254
SndWnd: 0000032768	MaxSndWnd: 0000032768
SndUna: 2045797254	rtt_seq: 2045797253
MaximumSegmentSize: 0000001411	DSField: 00
Round-trip information:	
Smooth trip time: 1.000	SmoothTripVariance: 1124.000
ReXmt: 000000000	ReXmtCount: 000000000
DupACKs: 0000015382	RcvWnd: 0000032768
SockOpt: 00	TcpTimer: 00
TcpSig: 14	TcpSel: 40
TcpDet: E0	TcpPol: 00
QOSPolicyRuleName:	
<b>RoutingPolicy: Yes</b>	
<b>RoutingTableName: 70hs</b>	
<b>RoutingRuleName: T0202</b>	
ReceiveBufferSize: 0000016384	SendBufferSize: 0000016384
ReceiveDataQueued: 0000000000	
SendDataQueued: 0000000000	

----

Client Name: RODOLFI4	Client Id: 000000CA
Local Socket: 9.12.4.48..1025	Foreign Socket: 9.12.4.202..1953
Last Touched: 21:56:58	State: Estabsh

```

BytesIn:          0000000000      BytesOut:         0687437550
SegmentsIn:      0062680271      SegmentsOut:     0184224483
RcvNxt:         2046673459      SndNxt:         2733177595
ClientRcvNxt:   2046673459      ClientSndNxt:   2733189645
InitRcvSeqNum:  2046673458      InitSndSeqNum:  2045740044
CongestionWindow: 0001237066    SlowStartThreshold: 0000016384
IncomingWindowNum: 2046706227    OutgoingWindowNum: 2733177595
SndWl1:        2046673459      SndWl2:        2733177595
SndWnd:        0000000000      MaxSndWnd:     0000032768
SndUna:        2733177595      rtt_seq:       2733176059
MaximumSegmentSize: 0000001399  DSField:       00
Round-trip information:
  Smooth trip time: 1.000      SmoothTripVariance: 0.000
ReXmt:         0000000204      ReXmtCount:    0000000000
DupACKs:       0000045256      RcvWnd:        0000032768
SockOpt:       00              TcpTimer:      22
TcpSig:        10              TcpSel:        C0
TcpDet:        E0              TcpPol:        02
QOSPolicRuleName:
RoutingPolicy: Yes
  RoutingTableName: 70hs
  RoutingRuleName: T0202
ReceiveBufferSize: 0000016384    SendBufferSize: 0000016384
ReceiveDataQueued: 0000000000
SendDataQueued:  0000012050
  OldQDate:      05/16/2007      OldQTime:      21:56:58
-----

```

## C.6 SC70 netstat -A command

*Example: C-6 SC70 netstat -A command*

```

RODOLFI @ SC70:/u/rodolfi>netstat -A -P 1953
MVS TCP/IP NETSTAT CS V1R9      TCPIP Name: TCPIP          17:58:56
Client Name: RODOLF13          Client Id: 000000BF
Local Socket: 9.12.4.202..1953  Foreign Socket: 0.0.0.0..0
  Last Touched:      15:15:47      State:             Listen
  BytesIn:          0000000000      BytesOut:         0000000000
  SegmentsIn:      0000000000      SegmentsOut:     0000000000
  RcvNxt:         0000000000      SndNxt:         0000000000
  ClientRcvNxt:   0000000000      ClientSndNxt:   0000000000
  InitRcvSeqNum:  0000000000      InitSndSeqNum:  0000000000
  CongestionWindow: 0000000000    SlowStartThreshold: 0000000000
  IncomingWindowNum: 0000000000    OutgoingWindowNum: 0000000000
  SndWl1:        0000000000      SndWl2:        0000000000
  SndWnd:        0000000000      MaxSndWnd:     0000000000
  SndUna:        0000000000      rtt_seq:       0000000000
  MaximumSegmentSize: 0000000536  DSField:       00
Round-trip information:
  Smooth trip time: 0.000      SmoothTripVariance: 1500.000
ReXmt:         0000000000      ReXmtCount:    0000000000
DupACKs:       0000000000      RcvWnd:        0000032768

```

SockOpt:	00	TcpTimer:	00
TcpSig:	00	TcpSel:	00
TcpDet:	C0	TcpPol:	00
QOSPolicyRuleName:			
RoutingPolicy:	No		
ReceiveBufferSize:	0000016384	SendBufferSize:	0000016384
ConnectionsIn:	0000000001	ConnectionsDropped:	0000000000
CurrentBacklog:	0000000000	MaximumBacklog:	0000000010
CurrentConnections:	0000000001	SEF:	100
Quiesced:	No		

----

Client Name: ROD0LFI3  
Local Socket: 9.12.4.202..1953  
Last Touched: 21:58:56  
BytesIn: 1935416832  
SegmentsIn: 0185143649  
RcvNxt: 3981156877  
ClientRcvNxt: 3981156877  
InitRcvSeqNum: 2045740044  
CongestionWindow: 0000005644  
IncomingWindowNum: 3981189645  
SndWl1: 3981155772  
SndWnd: 0000032768  
SndUna: 2046673459  
MaximumSegmentSize: 0000001411  
Round-trip information:  
Smooth trip time: 1.000  
ReXmt: 0000000000  
DupACKs: 0000017363  
SockOpt: 00  
TcpSig: 14  
TcpDet: E0  
QOSPolicyRuleName:

**RoutingPolicy: Yes**  
**RoutingTableName: 65h**  
**RoutingRuleName: T048**  
ReceiveBufferSize: 0000016384  
ReceiveDataQueued: 0000000000  
SendDataQueued: 0000000000

----

Client Name: ROD0LFI4  
Local Socket: 9.12.4.202..1027  
Last Touched: 21:58:56  
BytesIn: 0000000000  
SegmentsIn: 0068866782  
RcvNxt: 2045797254  
ClientRcvNxt: 2045797254  
InitRcvSeqNum: 2045797253  
CongestionWindow: 0000344273  
IncomingWindowNum: 2045830022  
SndWl1: 2045797254  
SndWnd: 0000032768  
SndUna: 1648988984  
MaximumSegmentSize: 0000001399  
Round-trip information:

Client Id: 000000C2  
Foreign Socket: 9.12.4.48..1025  
State: Estabsh  
BytesOut: 0000000000  
SegmentsOut: 0063042031  
SndNxt: 2046673459  
ClientSndNxt: 2046673459  
InitSndSeqNum: 2046673458  
SlowStartThreshold: 0000065535  
OutgoingWindowNum: 2046706227  
SndWl2: 2046673459  
MaxSndWnd: 0000032768  
rtt\_seq: 2046673458  
DSField: 00  
SmoothTripVariance: 1124.000  
ReXmtCount: 0000000000  
RcvWnd: 0000032768  
TcpTimer: 00  
TcpSel: 40  
TcpPol: 00

SendBufferSize: 0000016384

Client Id: 000000C6  
Foreign Socket: 9.12.4.48..1953  
State: Estabsh  
BytesOut: 3897220096  
SegmentsOut: 0197824762  
SndNxt: 1648988984  
ClientSndNxt: 1648988984  
InitSndSeqNum: 2046736183  
SlowStartThreshold: 0000016384  
OutgoingWindowNum: 1649021752  
SndWl2: 1648988984  
MaxSndWnd: 0000032768  
rtt\_seq: 1648972600  
DSField: 00

```

Smooth trip time: 1.000          SmoothTripVariance: 0.000
ReXmt:          0000000199      ReXmtCount:        0000000000
DupACKs:        0000037131      RcvWnd:           0000032768
SockOpt:        00              TcpTimer:          00
TcpSig:         10              TcpSel:            C0
TcpDet:         E0              TcpPol:            02
QOSPolicyRuleName:
RoutingPolicy:    Yes
  RoutingTableName: 65hs
  RoutingRuleName:  T048
ReceiveBufferSize: 0000016384   SendBufferSize:    0000016384
ReceiveDataQueued: 0000000000
SendDataQueued:   0000000000
-----

```

## C.7 pasearch -R command

*Example: C-7 pasearch -R command*

---

```

RODOLFI @ SC65:/SC65/tmp>pasearch -R -e * display polic rules and actions

```

```

TCP/IP pasearch CS V1R9          Image Name: TCPIP
Date:          05/16/2007        Time:  18:22:33
Routing Instance Id: 1179327310

policyRule:          T0202
Rule Type:           Routing
Version:             4           Status:              Active
Weight:              500000      ForLoadDist:         False
Priority:             500000      Sequence Actions:    Don't Care
No. Policy Action:   1
policyAction:        T0202
ActionType:          Routing
Action Sequence:     0
Time Periods:
Day of Month Mask:
First to Last:       11111111111111111111111111111111
Last to First:      11111111111111111111111111111111
Month of Yr Mask:   111111111111
Day of Week Mask:   1111111  (Sunday - Saturday)
Start Date Time:    None
End Date Time:      None
Fr TimeOfDay:       00:00        To TimeOfDay:        24:00
Fr TimeOfDay UTC:   00:00        To TimeOfDay UTC:    00:00
TimeZone:           Local
Routing Condition Summary:          NegativeIndicator: Off
IpSourceAddr Address:
FromAddr:           9.12.4.48
ToAddr:             9.12.4.48
IpDestAddr Address:
FromAddr:           9.12.4.202
ToAddr:             9.12.4.202

```



TrafficDescriptor:  
Protocol: TCP (6)  
SourcePortFrom 0 SourcePortTo 0  
DestinationPortFrom 0 DestinationPortTo 0  
JobName RODOLFI\* SecurityZone  
SecurityLabel

Routing Action: T0202  
Version: 4 Status: Active  
UseMainRouteTable Yes  
RouteTable: 70hs

RODOLFI @ SC65:/SC65/tmp>pasearch -R -T \* displau the route tables

TCP/IP pasearch CS V1R9 Image Name: TCPIP  
Date: 05/16/2007 Time: 18:24:29  
Routing Instance Id: 1179327310

Route Table: 70hs  
Version: 1 Status: Active  
IgnorePathMtuUpdate No Multipath UseGlobal  
DynamicXCFRoutes No

Route  
Destination:  
ipaddress 9.12.4.202  
First Hop:  
gateway\_addr 10.1.101.70  
link\_name IQDIOLNK0A016541  
MTU size 1500  
Replaceable No  
MaximumRetransmitTime 120.000  
MinimumRetransmitTime 0.500  
RoundTripGain 0.125  
VarianceGain 0.250  
VarianceMultiplier 2.000  
DelayAcks Yes

InstanceId: 1179327310  
LastPolicyChanged: Wed May 16 10:55:10 2007

---

Archived

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 517. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *z/OS Version 1 Release 2 Implementation*, SG24-6235
- ▶ *z/OS Version 1 Release 3 and 4 Implementation*, SG24-6581
- ▶ *z/OS Version 1 Release 5 Implementation*, SG24-6326
- ▶ *z/OS Version 1 Release 6 Implementation*, SG24-6377
- ▶ *z/OS Version 1 Release 7 Implementation*, SG24-6755
- ▶ *z/OS Version 1 Release 8 Implementation*, SG24-7265
- ▶ *Implementing REXX support in SDSF*, SG24-7419
- ▶ *z/OS Distributed File Service zSeries File System Implementation z/OS V1R7*, SG24-6580
- ▶ *Communication Server TCP/IP Implementation, Volume 4: Policy-Based Network Security*, SG24-7169
- ▶ *Sysplex eBusiness Security z/OS V1R7 Update*, SG24-7150

## Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS Planning for Installation*, GA22-7504
- ▶ *z/OS MVS Using the Subsystem Interface*, SA22-7642
- ▶ *z/OS MVS Setting up a Sysplex*, SA22-7625
- ▶ *z/OS MVS Planning: Operations*, SA22-7601
- ▶ *z/OS MVS System Messages, Volume 10 (IXC-IZP)*, SA22-7625
- ▶ *z/OS MVS System Codes*, SA22-7626
- ▶ *z/OS MVS System Commands*, SA22-7627
- ▶ *z/OS MVS Installation Exits*, SA22-7593
- ▶ *z/OS MVS Programming: Workload Management Services*, SA22-7619
- ▶ *z/OS MVS Programming: Assembler Services Guide*, SA22-7627
- ▶ *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608
- ▶ *z/OS MVS Programming: Assembler Services Reference, Volume 2 (IAARR2V-XCRLX)*, SA22-7607

- ▶ *z/OS MVS Programming: MVS Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC), SA22-7609*
- ▶ *z/OS MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDTINFO-IXGWRITE), SA22-7610*
- ▶ *z/OS MVS Program Management: Advanced Facilities, SA22-7644*
- ▶ *z/OS Security Server RACF Callable Services, SA22-7691*
- ▶ *z/OS Security Server RACF Command Language Reference, SA22-7687*
- ▶ *z/OS Cryptographic Services Integrated Cryptographic Services Facility Writing PKCS #11 Applications, SA23-2231*
- ▶ *z/OS Cryptographic Services System Secure Sockets Layer Programming, SC24-5901*
- ▶ *z/OS Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide, SA22-7520*
- ▶ *z/OS Cryptographic Services Integrated Cryptographic Service Facility Administrator's Guide, SA22-7521*
- ▶ *z/OS Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide, SA22-7522*
- ▶ *z/OS Communications Server: IP Configuration Reference, SC31-8776*
- ▶ *z/OS Communication Server: IP Configuration Guide, SC31-8775*
- ▶ *z/OS Communication Server: New Function Summary, GC31-8771*
- ▶ *z/OS Common Information Model User's Guide, SC33-7998*
- ▶ *z/OS Metal C Programming Guide and Reference, SA23-2225*
- ▶ *z/OS SDSF Operation and Customization, SA22-7670*
- ▶ *z/OS XL C/C++ Compiler and Run-Time Migration Guide for the Application Programmer, GC09-4913*
- ▶ *z/OS Language Environment Programming Reference, SA22-7562*
- ▶ *eServer zSeries Common Information Model User's Guide, SC33-7998*
- ▶ *z/OS SDSF Operation and Customization, SA22-7670*
- ▶ *z/OS MVS Program Management: Advanced Facilities, SA22-7644*

## Online resources

These Web sites are also relevant as further information sources:

- ▶ The message flood automation function is accompanied by a User's Guide which can be found at the following link:  
<http://publibz.boulder.ibm.com/zoslib/pdf/mfaguide.pdf>
- ▶ The CFSIZER web site to recalculate the size of CF structures in a CFLEVEL 15 Coupling Facility is:  
<http://www.ibm.com/servers/eserver/zseries/cfsizer/>
- ▶ Information on PKCS #11  
<http://www.rsa.com/rsalabs/node.asp?id=2133Description3>
- ▶ You can access the ShopzSeries Web site at:  
<http://www.ibm.com/software/shopzseries>

- ▶ Examples from the SBLIM OpenSource project and can be downloaded from:  
<http://www.sblim.org>
- ▶ CIM Metrics White Paper (DSP0141), which is available at the DMTF web page in the CIM White Paper section at the following URL:  
<http://www.dmtf.org>
- ▶ The URL for the DFSMSrmm wizard is:  
<http://www.ibm.com/servers/eserver/zseries/zos/wizards/dfsms/rmmv1r9/>
- ▶ The Pegasus CIM Server 2.5.3 website:  
<http://www.openpegasus.org>
- ▶ For more information on the zIIP configuration, see and perform a search on zIIPs:  
<http://www.ibm.com/support/techdocs>
- ▶ A whitepaper is available at the following link called Capacity Planning for zIIP Assisted IPsec which describe in details how to plan for zIIP capacity.  
<http://www.ibm.com/support/docview.wss?rs=852&uid=swg27009459>

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

Archived

# Index

## Symbols

\$C Job command 356  
\$DSPL performance 352  
\$JCT 361  
\$JD DETAILS(STORAGE) 354  
\$JD HISTORY(STORAGE) 354  
\$SSPL command 352  
\$T TRACE 357  
\$TRACE facility 357  
&SYSCONE 421  
\*START,DC 363  
\_\_far\_jump() 489  
\_\_iew\_api.h 345  
\_\_le\_traceback() 489  
\_UNIX03\_SOURCE 158  
\_UNIX03\_THREADS 158

## Numerics

3592 Model E05 430  
3-way audit 403

## A

Abend 878 reduction 389  
ABEND code 353  
access bias 379  
AD dsname 414  
AddDS 433  
ADDUSER 182  
AES 195  
aggrgrow option 287  
ALIGN option  
    HEAPPOOLS 483  
Alternate Library for REXX 19, 25  
ALTUSER 182  
APAR OA08688  
    REALLOCATE enhancement 49  
APAR OA16731 363  
APAR OA17055  
    coexistence CF duplexing 37  
    RMF and CF 40  
    XCF CF level 41  
APAR OA17070  
    coexistence RMF 40  
APAR OA17514  
    console message flood 7  
    message flood automation 106  
APAR OA17685  
    coexistence CF maintenance mode 47  
APAR OA17827 343  
APAR OA18244  
    RMF SPE blocked workloads 133  
APAR OA18531  
    IWMSRSRS enhancements 141

APAR OA20186 238  
APAR OA20314  
    zNALC 34  
    ZNALxxxx LPAR name 34  
APAR OW55574 362  
APAR PK29028  
    LE coexistence 479  
ARC1MIGR 389  
archive libraries 347  
ARCRPEXT 392  
ARM 313  
ARM element name  
    CFZ\_SRV\_ 323  
ARTQSRV program 469  
AS DSNAME 414  
ASCII 176  
ATRQSRV Batch Utility 468  
ATRRRS Batch Support 468  
ATRSRV interface 473  
AT-TLS facilities  
    CIM server 313  
Automatic Restart Manager  
    CIM 313  
    CIM Server 20  
AUTOMOVE consistency 276  
AV DSNAME 414  
available spool records (ASR) 363  
AXR address space 310  
AXREXX  
    assembler macro interface 240  
AXREXX cancel service 369  
AXREXX macro service 239, 242  
AXRPSTRT procedure 238  
AXRxx parmliib membe 241

## B

BACK 389  
BACKOUT 469  
Basic Access Methods (BAM) 376  
best-fit allocation 383  
Binder C/C++ 345  
Binder C/C++ API 345  
Binder C/C++ API DLL file 346  
Binder C/C++ API header file 345  
Binder C/C++ API side-deck file 346  
Binder control statement 343, 347  
Binder data set 344  
Binder fast data access services 348  
Binder module map toleration 343  
Binder option 342–343  
blocked workload  
    RMF support 133  
BLWLINTHD  
    blocked workloads 132

- BLWLTRPCT
  - blocked workloads 132
- bpxmtext utility 482
- BPXPRMxx 433
- BRIF 164
- BRINGIN SYSEVENT 128
- BROWSE 164
- Buffer 437

## C

- CALLRTM 356
- CANCEL 356
- CANCEL SYSEVENT 128
- CBROAMxx 385
- CBROAMxx parmlib member 386
- CBRSMR19 388
- CBRXLCS 403
- CCA 55
- CD NEWDSNAME 414
- CDS 406
- CDSA 196
- CDSB 389
- CDSID parmlib option 407
- CEE3MC2 callable service 486
- CEE5DLY 486
- ceebldtx utility 483
- CEEDLYM 487
- CEEDUMP option 478
- CEEDUMP report 478
- CEEHGOTO 488
- CEELRR 490
- CEETBCK 488
- CELQPIPI 487
- CELQPIPI interface 487
- Certificate Authority (CA) 190
- certificate revocation lists (CRLs) 191
- CF duplexing 36
- CFCC 36
- CFCC multitasking enhancements 37
- CHANGE 343
- CHECKROUTINE 367
- CIM 20, 312
- CIM client application 314
- CIM Client for Java 313
- CIM client for Java API 319
- CIM data instrumentation 318
- CIM monitoring clients 20
- CIM provider 315–316
  - DFSMSrmm 412
- CIM security 320
- CIM server
  - logging facility 313
- CIM server runtime 319
- CIM/XML over HTTP 314
- cimconfig command 325
  - logLevel parameter 326
- CIMSERV RACF profile 322
- cimsub 319
- cimsub command
  - CIM indications 326

- CIM-XML over HTTP 314
- CIM-XML over HTTP protocol 314
- CLASSPATH environment variable 186
- CLER run-time option 485
- CLIST Enhancements 424
- CNZZCMXT 113
- CNZZCMXT command exit 123
- CNZZCMXT command exit routine 114
- CNZZMFxx
  - SYS1.SAMPLIB member 115
- CNZZVXT1 or CNZZVXT2 110
- Command Table Utility 178
- COMMIT 469
- COMMNDxx parmlib member
  - MSGFLD commands 123
- Common Cryptographic Architecture 55
- Common Data Security Architecture 183, 196
- Common Information Model 20, 312
- Common Information Model (CIM) 409
- Communications Server
  - policy-based routing 13
- COMPARE command 169
- CONSOLxx parmlib member
  - INIT UEXIT(Y) 113
  - MPF keyword 110
- CONTINUE 412
- continue\_information 426
- CONTINUOUS 381
- Continuous 382
- Continuous Preferred 382
- CONTINUOUS PREFERRED. 381
- control facility control code
  - CFLEVEL 15 36
- COPY command 169–170
- Coupling Facility
  - maintenance mode 46
- Coupling Facility log stream
  - SMF recording 79
- CP Assist for Cryptographic Function (CPACF) 195
- CPACF 195
- CPOOL service
  - GRS latch support 103
- CREATE 408
- CREATE command 169, 171
- CRL 192
- Crypto Express2 (CE2) 202
- cryptographic keys data set 55
- Cryptographic Services Security Level 3 FMID 189
- Cryptographic Services System SSL FMID 189
- CRYPTOZ
  - new RACF class 56
- CRYPTOZ resource class 55, 59
- CSDID 406
- Ctrace 436
- CV DSNAME 414

## D

- D A,ZFS command 288
- D CF command
  - CFLEVEL 15 40



- D IPLINFO command
  - LICENSE=zNALC 34
- D LOGGER,ST,REC command 72
- D MF,MSGRATE command 121
- D OMVS,W command 287
- daily statistics records (DSR) 390
- DASDONLY log streams
  - SMF recording 7
- DASD-only log streams
  - SMF recording 79
- Data Facility Storage Management Subsystem
  - DFSMS 375
- dataclass 430
- DayChart 457
- DB2 performance
  - GRS enhancements 100
- DCBE 377
- DDS 42
- DEBUG mode 369
- decimal floating-point 253
- defined capacity 144
- DELETERM 469
- DES 195
- DFHSMrmm
  - QUERY ACTIVE command 394
- DFHSMrmm QUERY ACTIVE 394
- DFP 253
- DFSMSshm 388
  - REPORT command 392
- DFSMSrmm 393
- DIAGxx parmlib member
  - ISGERQA parameter 100
- Dialog Tag Language (DTL) 179
- DIGTCERT 185
- DIGTNMAP 185
- DIGTRING 185
- Direct Access Storage Device (DASD) 388
- Direct Optimized (DO) 379
- Direct Weighted (DW) 380
- Directory List Actions panel 167
- disk 384
- DISPLAY CF command 39
- DISPLAY LOGGER command 72
- DISPLAY MSGFLD command 123
- DISPLAY PROG,EXIT command
  - GRS exits 101
- Distinguished Name 198
- Distributed Data Server 448
- Distributed Management Task Force 312
- DLL diagnostics 272
- DMTF 312
- DSTORE 399
- DUMP 389
- dump job (DJ) facility 362

## E

- Eclipse-based graphical user interface 21
- edcmtext utility 482
- EDG@CONT 427
- EDG0237E 408

- EDG3012I 427
- EDG3025I 427
- EDG3203I 427
- EDG3360E 425
- EDGGDSNM 419
- EDGHSKP 398
- EDGINERS 393
- EDGRMMxx 420
- EDGRMMxx parmlib member 420–421
- EDGRRPTE 423, 429
- EDGSPLCS 398
- EDGSPLCS utility 404
- EDGUTIL 398, 401
- EDGUTIL message 408
- EDIF 164
- EDIT 164
- Edit Entry Panel 164
- EDIT macro commands 174
- EInterval 371
- ENF signal 65
  - cancel AXR 238
- ENQ request
  - ISPF z/OS UNIX files 167
- Enterprise Encrypted Format 2 (EEFMT2) 430
- Enterprise Format 1 (EFMT1) 430
- Enterprise Format 2 (EFMT2) 430
- enumerateInstanceNames 411
- EOV new volume extend 383
- errno2 support 481
- EXEC 367
- ExecName 371
- exit 31 361
- exit 42 361
- exit 45 361
- EXPROC 398
- EXPROC command 399
- Extended OSE buffer numbers toleration 363
- Extent Reduction 392
- extern functions 255–256
- EXTRDCTN 392

## F

- F AXR command 241
- F DFRMM,M=xx 396
- F OAM,DISPLAY command 387
- F OAM,RESTART 385
- F ZFS,ABORT command 288
- F ZFS,HANGBREAK command 288
- F ZFS,QUERY,THREADS command 287–288
- FACILITY class 184
- fdlibm math functions 482
- FILEXFER 175
- filters 357
- FIXED=USER 377
- FLD name 294
- FMID 189
- FMID HBB77SR
  - System REXX 25
  - System REXX with V1R8 238
- FMID JDZ118E

DFSMsdfp English panels 25  
FMID JPG290B  
  CIM clients 25  
FORCE 356  
FSRSTAT program 391  
FSRTYPE 391  
Functional Statistics Record (FSR) 390

## G

GENASM option 255  
GETFILE 443  
Global Resource Serialization (GRS) 408  
GLOBALCONFIG statement  
  zIIP IPsec function 202  
GLOBALCONFIG ZIIP IPSECURITY 203  
GPMSRVxx PARMLIB member  
  DDS settings 42  
group capacity limit 144  
group capacity limits 150  
GRS complex 97  
GRS latch function 103  
GRSRNLxx parmlib member 408  
gsk\_attribute\_set\_callback 193  
gsk\_attribute\_set\_enum 192  
GSK\_CRL\_SECURITY\_LEVEL 192  
gsk\_reset\_callback 193  
gsk\_validate\_hostname 194  
gsk\_validate\_server 194  
GSKCMS\_VALIDATE\_HOSTNAME\_CN 194  
GSKCMS\_VALIDATE\_HOSTNAME\_CN\_ONLY 194  
GSKCMS\_VALIDATE\_HOSTNAME\_DNS 195  
GSKCMS\_VALIDATE\_HOSTNAME\_DNS\_ONLY 195  
gskkyman 57, 190  
gskkyman command 57  
GTTTERM macro 16  
gxlpInit 462  
gxlpParse 462  
gxlpTerminate 462

## H

Hardware Configuration Manager 182  
HCM 182  
HCPT390 189  
HEAPOOLS runtime options 483  
HiperSockets 228  
HLL environment 487  
host command environment  
  SDSF REXX 293  
HSM DFSMSdss address spaces 390  
HZZADDCK macro 368

## I

IBM System z9  
  group capacity limit 144  
IBM WebSphere Application Server 413  
IBM WebSphere Developer Debugger for System z 21  
IBM WebSphere Developer for System z 262  
IBM-1074 185

IBMmrm\_Control 410  
IBMmrm\_PolicyRule 410  
IBMmrm\_Product 410  
IBMmrm\_SearchOperands 411  
IBMzOS\_LogicalDisk  
  CIM 313  
IBMzOS\_LogicalDisk class 331  
IHPWX11 182  
iconv() 22, 477  
ICSF 195  
ICSF installation options  
  TKDSN option 57  
ICSF started procedure 59  
IdenTrust 196  
IEAOPTxx parmlib member  
  CPU management constants 132  
  PROJECTCPU 203  
IEASYSxx parmlib member  
  LICENSE parameter 34  
IEAVMXIT exit  
  message flood automation 108  
IEAVMXIT installation exit 108  
IEAVMXIT message exit  
  existing installed exit 111  
IEC161I 379  
IEFU29L  
  SMF log stream dump exit 91  
IEFUSI exit  
  MEMLIMIT 348  
IEWBIND 345  
iewbndd.so 346  
iewbndd.x 346  
IEWBUFF 345  
IFASMF DL  
  new SMF dump program 91  
IFASMF DL utility  
  new SMF dump program 88  
IKJTSOEV  
  dynamic TSO service 240  
IKJTSOxx parmlib member  
  TRANSREC statement 442  
IMPORT 347  
INFO 343  
initACEE 185  
Initial Access Response Seconds (IARS) 384  
INMLSIZE 441  
INMX033I 443  
INMX034I 441  
installation wizard 422  
Integrated Cryptographic Service Facility (ICSF) 195  
Intelligent Resource Director 150  
IntervalChart 457  
invokeMethod() 411  
IOEAGFMT utility 283  
  ALTER authority 284  
IOEAGSLV utility 283  
  UPDATE authority 284  
IPSEC 196  
IPSec 202  
IPSEC statement 204

- iQDIO links 226
- IRD 150
- IRRPHEX 183
- IRRSDL00 183
- IRRSDL64 183
- ISFACT 300
- ISFACT command 293
- ISFCALLS command 293
- ISFEXEC command 293–294
- ISGNQXIT EQDQ exit 102
- ISGNQXITFAST exit 102
- ISHELL 164
- ISPF function 164
- IWM4SLI service 129
- IWMWSYSQ service 143
- IXCM2APU utility
  - create log streams 79
- IXCMIAPU utility
  - LIST LOGSTREAM request 72

## J

- Java security API 186
- JCL AMP keyword, MSG=SMBBIAS 379
- JCPT391 189
- JCPT39JCryptographic Services Japanese FMID 189
- JCT 361
- JES2 \$TRACE facility 357
- JES2 exit 361
- JES2 exit 8 361
- JES2 initialization statement 357
- JES3 64k OSE buffers support 362
- JES3 command 363
- JES3 global 362
- JES3 local 362
- JESJOBS 354
- JESSPOOL 354
- JSec 186

## K

- K M,UEXIT command
  - Message Flood Automation 113
- K M,UEXIT(Y) command
  - activate IEAVMXIT 122

## L

- L2DATACLASS 385
- L2TAPEUNITNAME 385
- Label Anomaly processing 432
- Large format data sets 439
- large format data sets 440
- LDAP 186, 191, 198
- LEDATA IPCS verbexit command 269
- LF 177
- LF command
  - realign ascii data 177
- LibraryCenter 309
- LIBSERV 404
- LICENSE=ZNALC parameter 34

- List Data Set Information (LISTDSI) 442
- LISTCONTROL 421
- Locale 372
- Lock\_Release 437
- Lock\_Request 437
- Lock\_Result 437
- Locking 435
- logical disk volumes
  - CIM 331
- LOGINFO 469
- Long Running Tasks 394
- Long-term page fixing 377
- LPAR LIC 145
- LPAR Trend report 454
- LPAR weights 145

## M

- maintenance mode 46
- masks 417
- media types 430
- MEND(SMSTAPE) 403
- message flood automation
  - SPE 106
- message processing facility (MPF)
  - message flood processing 108
- Message Rate Monitoring function 121
- METAL 460
- Metal C Runtime Library 25
- Metal C runtime library 257
- METAL option 254
- MethodProvider2 411
- MIGR 389
- MIXC keyword 179
- MLACTIVE 353, 355
- MLNAMES 434
- MODMAP 342
- mount command 436
- MOVE command 169, 172
- MPFLSTxx parmlib member
  - CNZZCMXT 113
  - USEREXIT parameter 108
- MSGFLDxx parmlib member
  - message flood automation 116
  - statement types 115
- msys for Operations 24
- MULTACC 376
- multiple TCPIP stacks 432
- Multitasking of utilities 398
- MULTSDN 376, 378
- MVS command 356
- MVS JCL DSNAME 414

## N

- NALC 30
- Name-Hiding 434
- Network File System (NFS) 432
- New Application License Charges 30
- new SYSREXX support for V1R8 238
- NFS 432

NFS Client 436  
NFS client enabling utility 438  
NFS control data set 434  
NFS Server 436  
NFS Server tasks 432  
NFS V4 protocol 435–436  
NFS v4 Protocol 435  
Nopref 382  
normal allocation 383  
NOSECLEVEL 422

## O

OAM Operator Commands 386  
OAM storage management component (OSMC) 384  
OAM Sublevel (OSL) 384  
Object Access Method (OAM) 384  
OBROWSE 164  
ODLOCFL column 387  
OEDIT 164  
OMVS 164  
OPENFILE 443  
OpenPegasus CIM Server 2.5.3 413  
OpenPegasus CIMOM 409  
OpenPegasus CVS Repository 321  
OPERLOG  
    System Logger 71  
optical 384  
Origin 372  
OSE 362  
OSREQ macro 384  
OUTLIM operand 441  
Output Scheduling Element (OSE) 362  
Overview Report spreadsheet 457

## P

PAGENT configuration 204  
Password phrase 182  
PDF editor 169  
PDF installation-wide data set allocation exit 176  
Pegasus CIM Server 412  
Pegasus CIM Server 2.5.3 412, 517  
Personal Data Set List panel 168  
personal data set lists 168  
PHRASE 182  
PIMA 462  
PKCS #11 54–55, 190  
PKCS #11 tokens 54  
PKCS #7 190  
PKI Services 196  
PKIX 196  
Postprocessor I/O Queuing Activity Report 446  
PreInit 487  
PreInit driver program 487  
PreInit environment 269  
PRINTDS 442  
PROFILE PREFIX 416  
public 436  
Public Key Infrastructure for X.509 196  
public keys data set 55

public-key cryptography 191  
PUTFILE 443

## Q

Qualified Application  
    zNALC 30  
QUERY ACTIVE command 395  
quiesce the DFSMSrmm subsystem 397  
quoted 417

## R

R\_datalib 183, 185  
R\_PKIServ 199  
RACDCERT 183, 185  
RACEXITS 183  
RACF class  
    WBEM 319  
RACF Data Labeling 434  
RCVR 389  
RDATALIB 184  
REALLOCATE command 46  
Record length 165  
Record length field  
    ISPF panel 165  
Redbooks Web site 517  
    Contact us xvii  
REFACTD command  
    personal data set list 168  
registration services  
    RRS 471  
REJECT 420  
REMOVINT 469  
REPLACE 343  
REPLACE command 169, 173  
REPORT 17 423  
Report Class Periods 454  
report generator 419  
REPORT17 429  
RESET 177  
Resource Measurement Facility 445  
REST 389  
restart DFSMSrmm 396  
Return Priority (RP) 392  
REXX 183, 291, 366  
REXX EXECIO 443  
REXX HZSMMSGEN procedure 370  
REXX on zSeries Library 19  
REXX variables  
    SDSF support 293  
REXXC procedure 370  
REXXHLQ 367  
REXXIN data set 238  
REXXIN=YES/NO 367  
REXXTIMELIMIT 367  
REXXTSO=YES/NO 367  
RFC 2246 193  
RFC 2818 194  
RMF Distributed Data Server 42  
    CIM client/server 314

RMF enhancements for FICON 445  
 RMF metrics  
     CIM client/server 314  
 RMF Monitor III Data Portal 445  
 RMF PM Java client 448  
 RMF Spreadsheet Reporter 451  
 RMF Spreadsheet Reporter Enhancements 445  
 RMINFO 469  
 RMM CIM classes 409  
 RMM CIM provider 409  
 RMM CIM provider v1.9 413  
 RMM SUBCOMMANDS 414  
 RMM Web Service 428  
 RPCSEC\_GSS Security 435

## S

S/MIME 196  
 SOC4 353  
 S1E0 353  
 SAF or SAFEXP 438  
 SBLIM project 328  
 Scheduler Work Block Text Unit (SWBTU) 353  
 SCRT reporting period 147  
 SDBM 198  
 SDBM backend 186  
 SDSF 291  
 SDSF CK 371  
 SDSF display 390  
 SDSF source modules and macros 305  
 SearchVolume 429  
 SECLABEL 353, 355  
 Secure Sockets Layer 189  
 SEND command 355  
 Sequential Optimized (SO) 380  
 Sequential Weighted (SW) 380  
 SET MPF= command  
     message flood exit 113  
 SET MPF=xx command  
     active message flood commands 122  
 SET SMF=XX command  
     SMF log streams 85  
 SETLOGR FORCE command 73  
 SETMF command 125  
 SETMF MONITOROFF command 121  
 SETMF MONITORON command 121  
 SETMF ON command  
     activate message flood 115  
 SETOAM statement 386  
 SETOAMxx keywords 386  
 SETROPTS 182  
 SETROPTS MLNAMES 435  
 SETROPTS NOMLNAMES 435  
 SETROPTS REFRESH 185  
 SFI 381  
 SHA-1 195  
 SHA-256 195  
 Shared parmlib support 420  
 slash (/) command 298  
 SMBVSP keyword 379  
 SMF dump program

IFASMF DL 91  
 SMF record 92  
     z/OS UNIX 5  
 SMF record type 72-3  
     blocked workloads 135  
 SMF record type 88 72  
 SMF type 70 record  
     group capacity limits 150  
 SMF type 89 record  
     IPL with zNALC 34  
 SMF type 99 record  
     group capacity data 150  
 SMFPRMxx parmlib member  
     LSNAME parameter 84  
     SMF recording 83  
 soft cap 144  
 soft shutdown 280  
 Sort capability 449  
 SOURCE 176  
 Special REXX variables 295  
 SSI 11 353  
 SSI 6 356  
 SSI 70 353  
 SSI 71 353  
 SSI 75 355  
 SSI 79 355  
 SSI 80 355  
 SSI 81 356  
 SSL 189, 196  
 SSRV  
     new trace for ENQ/RESERVE/DEQ 103  
 stacks and heaps 432  
 Standard 382  
 storage class (SC) 384  
 storage facility image 381  
 STORAGEGROUP 429  
 stringprep 436  
 striping allocation 383  
 Stunted volume processing 352  
 SUBLEVEL 387  
 subsystem interface (SSI) 352  
 SUPERUSER.FILESYS.PFSCtl profile 283  
 Sustained Data Rate (SDR) 384  
 SUSv3 156  
 SVC 99 176, 356  
 swapldb 434  
 swapmldb 434  
 SWITCH SMF command  
     dumping log stream data 91  
 syntax of the adds 434  
 syntax of the frees 434  
 syntax of the swap 434  
 SYS1.LINKLIB  
     IEAVMXIT exit 112  
 SYS1.MAN data sets 7  
 SYS1.PROCLIB  
     AXRPSTRT procedure 238  
 SYS1.SAMPLIB  
     Health Checker REXX execs 367  
     member CSFTKDS 57

SYS1.SAMPLIB(ERBWBE1  
     RMF CIM client monitoring application 336  
 SYS1.SAXREXEC 370, 372  
     Health checker REXX execs 367  
 SYS1.SIEALNKE 57  
 SYSINFO 469  
 SYSLMOD 344  
 syslog daemon  
     CIM logging 326  
 Sysplex Distributor  
     WLM routing services 136  
 Sysplex wide reports 448  
 sysplex wide reports  
     RMF 448  
 SYSPLEXTKDS option  
     ICSF 62  
 SYSPLEXTKDS(YES,FAIL(xxx))  
     ICSF 62  
 SYSREXX 366  
 SYSSEQDSNTYPE 442  
 System Logger  
     SMF recording 7  
 system managed buffering 379  
 System Managed Buffering (SMB) 379  
 System REXX 25, 183, 310  
 System service 356  
 System SSL 183, 189  
 System SSL API 192  
 System SSL environment variable 192  
 System SSL runtime services 58  
 system-managed duplexing rebuild 36  
 SYSUSED 442  
 SYSZRMM MASTER.RESERVE 407

## T

tape and DASD data sets  
     QSAM 377  
 tape configuration database (TCDB) 402  
 tape data sets  
     BSAM 377  
 tape sublevel 1 (TSL1) 384  
 tape sublevel 2 (TSL2) 384  
 TCBuffers 455  
 TCOverview 455  
 TCP/IP 189  
 TCSystems 455  
 TDES 195  
 tilde 164  
 TKDS 55, 57  
 TLS 189  
 TOKEN 300  
 token data set 55  
 TRACE 357  
 Traceback  
     LE dump 479  
 Traditional MSUs 33  
 Transport Layer Security 189  
 TRANSREC statement  
     OUTLIM operand 441  
 traversing 410

TrendChart 457  
 TS1120 Tape Drive 28

## U

UEXIT(Y) parameter 113  
 Unicode conversion services 477  
 UNIXPRIV class 283  
 unquoted 417  
 UNREGISTER RM command 471  
 UNREGRM 469  
 UPDATE 408  
 URINFO 469  
 UTF8 185

## V

VCINOUT 404  
 verbexit command 268  
 Verbose 372  
 VERIFY(SMSTAPE), MEND 403  
 VERIFY/MEND(VOL) 402  
 VIEW 164  
 View Entry Panel 164  
 Virtual Private Network (VPN) IP Security (IPsec) 202  
 vital record specification (VRSEL) 418  
 VLPOOL 420  
 volume statistics records (VSR) 390  
 VPN 183, 196  
 VRSEL 399  
 VRSEL processing data set names 418

## W

WARN/NOWARN  
     TSO TRANSMIT 441  
 WBEM 313  
 WBEM class 322, 326  
     RACF class for CIM 322  
 WHO command 293  
 WMINFO 469  
 Workload Activity 451  
 Workload Activity report 454  
 Workload License Charges 144  
 WOSE 363

## X

X.509 190  
 XCF Activity 451  
 XCF Activity report 455  
 XCF Path performance 455  
 XCF Transport Class 455  
 XL C compiler 253  
     METAL option 253  
 xlc command 255  
 XML parser 460, 465  
 XML System Services 459–460, 465  
 XP-LINK 464  
 XPLINK enhancements 489

## Z

z/OS CIM Server v1.9 413  
z/OS Cryptographic Services 196  
z/OS NFS RPCSEC 432  
z/OS RMF Distributed Data Server 42  
z/OS Unicode Conversion Services 177, 185  
z/OS UNIX Directory List Utility 164  
z/OS XML parser 465  
z/OS XML Toolkit 460  
z/OS.e  
    planned withdrawal 30  
z890 195  
z9 BC 195  
z9 EC 195  
z990 195  
zAAP 465  
    XML System Services 465  
zAAP and zIIP 452  
zfsadm define  
    dynamic grow 286  
zNALC 30

Archived





**Redbooks**

# **z/OS Version 1 Release 9 Implementation**

(1.0" spine)

0.875" x 1.498"

460 x 788 pages







# z/OS Version 1 Release 9 Implementation



**JES2, JES3, GRS, SMF,  
WLM, z/OS UNIX, zFS**

**Health Checker, SDSF,  
System REXX, Binder,  
DFSMS**

**Message flood, XML,  
CIM, z/OS base**

This IBM Redbooks publication describes the functional enhancements to IBM z/OS for Version 1 Release 9 (z/OS V1R9). These enhancements help installations to install, tailor, migrate, and configure z/OS V1R9.

IBM z/OS Version 1 Release 9 overview  
Installation and migration to z/OS V1R9  
Coupling Facility enhancements  
ICSF support for PKCS #11  
Allocation dynamic storage improvements  
System Logger enhancements  
SMF recording to log streams  
Message Flood Automation  
C/C ++ enhancements  
Security enhancements  
z/OS Communication Server  
New faces of z/OS  
System REXX for z/OS  
z/OS XL C Metal option  
z/OS UNIX System Services  
SDSF enhancements  
Program management enhancements  
JES2 and JES3 enhancements  
IBM Health Checker for z/OS  
Large format data sets  
GRS, WLM, ISPF, DFSMS, RMF, XML, and RRS enhancements  
Language Environment

**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](http://ibm.com/redbooks)**

SG24-7427-00

ISBN 0738488607