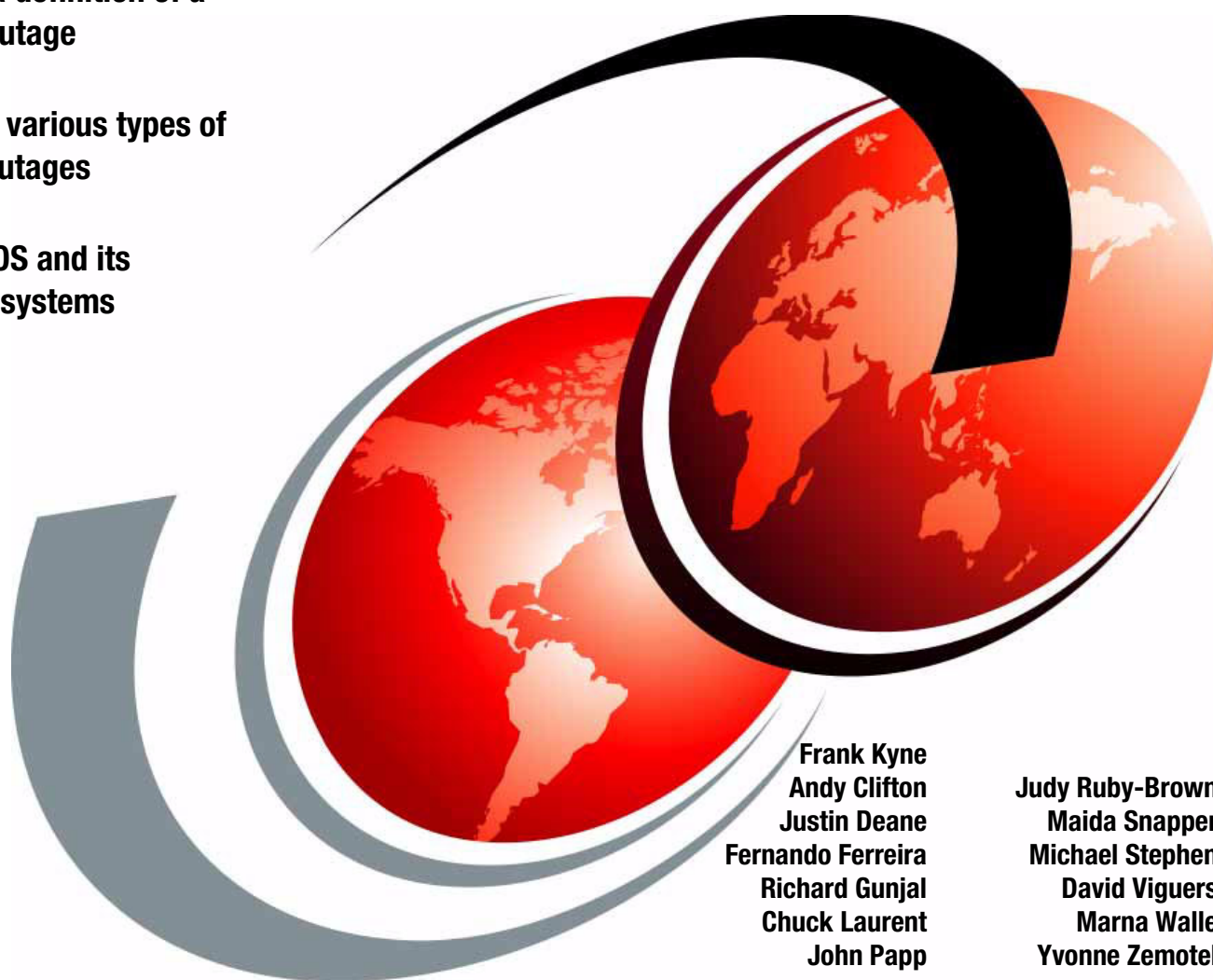


# Improving z/OS Application Availability by Managing Planned Outages

Provides a definition of a planned outage

Describes various types of planned outages

Covers z/OS and its major subsystems



Frank Kyne  
Andy Clifton  
Justin Deane  
Fernando Ferreira  
Richard Gunjal  
Chuck Laurent  
John Papp

Judy Ruby-Brown  
Maida Snapper  
Michael Stephen  
David Viguers  
Marna Walle  
Yvonne Zemotel

# Redbooks





International Technical Support Organization

**Improving z/OS Application Availability by Managing  
Planned Outages**

December 2014

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

#### **First Edition (December 2014)**

This edition applies to Version 2, Release 1 of z/OS (product number 5650-ZOS) and Version 1 of z/OS (product number 5647-A01), Version 5 Release 1 of CICS/TS (product number 5655-Y04), Version 11 Release 1 of DB2 (product number 5615-DB2), Version 13 Release 1 of IMS (product number 5635-A04), Version 7 Release 1 of WebSphere MQ (product number 5655-R36) and Version 8 Release 5 of WebSphere Application Server (product number 5655-W65)

**© Copyright International Business Machines Corporation 2014. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	xvii
Trademarks .....	xviii
<b>IBM Redbooks promotions</b> .....	xix
<b>Preface</b> .....	xxi
Authors .....	xxi
Now you can become a published author, too! .....	xxiii
Comments welcome .....	xxiv
Stay connected to IBM Redbooks .....	xxiv
<b>Chapter 1. Planned outage considerations</b> .....	1
1.1 What is a planned outage? .....	2
1.2 System availability or application availability? .....	3
1.3 Planned outage avoidance or control? .....	4
1.4 Differences between planned and unplanned outages .....	5
1.5 Reasons for planned outages .....	5
1.6 What this book covers .....	6
1.7 Layout of this book .....	7
1.8 Live Guest Relocation .....	8
<b>Chapter 2. Project planning and user experiences</b> .....	9
2.1 Project planning considerations .....	10
2.2 Setting the rules .....	10
2.3 Teams to involve .....	11
2.4 Researching and developing a strategy .....	13
2.5 Changes and how to control them .....	14
2.6 How frequently should I IPL or restart my subsystems? .....	16
2.7 Systems Management considerations .....	17
2.8 User experiences .....	18
2.8.1 Batch schedulers .....	18
2.8.2 Session manager products .....	18
2.8.3 Tasks using Getmain and Freemain .....	18
2.8.4 Log files and data sets .....	19
<b>Part 1. z/OS components</b> .....	21
<b>Chapter 3. z/OS enhancements: Availability and diagnostics</b> .....	23
3.1 List of z/OS enhancements .....	24
3.2 PARMLIB management for availability .....	27
3.2.1 Syntax checker: Language Environment .....	28
3.2.2 Syntax checker: z/OS UNIX .....	28
3.2.3 Syntax checker: TSO/E .....	28
3.2.4 Syntax checker: BCP PARMLIB Processor .....	28
3.2.5 Syntax checker: JES3 .....	29
3.2.6 Syntax checker: Other products .....	30
3.3 Dynamic exit support for availability .....	30
3.3.1 Dynamic exit support for DADSM IGGPRE00 and IGGPOST0 exits .....	31
3.3.2 Dynamic exit support for CSVLLIX1 and CSVLLIX2 (LLA exits) .....	32

3.4 Scalability for availability . . . . .	32
3.4.1 Extended Address Volume (EAV) . . . . .	33
3.4.2 Base Control Program Internal Interface (BCPii) . . . . .	34
3.5 Diagnostics for availability . . . . .	35
3.5.1 IBM Health Checker for z/OS . . . . .	35
3.5.2 Runtime Diagnostics component . . . . .	39
3.5.3 Runtime Diagnostics with OMVS latch contention . . . . .	41
3.5.4 Enhanced SLIP DUMP for non-dispatchable (MAXNDSP and DEFERTND) . . . . .	42
3.5.5 Verify processor type and number with CONFIG . . . . .	43
3.5.6 Diagnosing catalog contention issues . . . . .	44
3.5.7 Enhanced D GRS,ANALYZE command . . . . .	45
3.5.8 Restartable SMSPDSE1 address space . . . . .	46
3.5.9 Improved IPL-time support for corrupted LNKST PDSE data sets . . . . .	48
3.5.10 Enhanced commands to identify corrupted PDSE control blocks . . . . .	48
3.5.11 IEBPDSE utility to check for damaged PDSEs . . . . .	49
3.5.12 Dynamically start and stop VSAM record management trace . . . . .	50
3.5.13 Detecting a single point of failure programmatically ( <b>IOSSPOF</b> ) . . . . .	51
3.5.14 Hardware Instrumentation Service (HIS) detects change in CPU speed . . . . .	53
3.5.15 Predictive Failure Analysis (PFA) . . . . .	55
<b>Chapter 4. z/OS enhancements: Base Control Program . . . . .</b>	<b>57</b>
4.1 List of Base Control Program enhancements . . . . .	58
4.2 Change nearly any value in ALLOCxx without an IPL . . . . .	59
4.2.1 The issue . . . . .	59
4.2.2 How this is addressed in z/OS 1.11 . . . . .	59
4.2.3 Restrictions or considerations . . . . .	59
4.2.4 Rollback . . . . .	59
4.2.5 Further information . . . . .	60
4.3 More effective use of space in ECSA . . . . .	60
4.3.1 The issue . . . . .	60
4.3.2 How this is addressed in z/OS 1.10 . . . . .	60
4.3.3 How this is addressed in z/OS 1.13 . . . . .	61
4.3.4 Restrictions or considerations . . . . .	62
4.3.5 Rollback . . . . .	62
4.3.6 Further information . . . . .	62
4.4 PROG enhancement: Dynamic LNKST . . . . .	62
4.4.1 The issue . . . . .	63
4.4.2 How this is addressed in z/OS 1.12 . . . . .	63
4.4.3 Restrictions or considerations . . . . .	64
4.4.4 Rollback . . . . .	64
4.4.5 Further information . . . . .	64
4.5 PROG enhancement: Dynamic LPA . . . . .	64
4.5.1 The issue . . . . .	64
4.5.2 How this is addressed in z/OS 1.12 . . . . .	64
4.5.3 Restrictions or considerations . . . . .	65
4.5.4 Rollback . . . . .	65
4.5.5 Further information . . . . .	65
4.6 PROG enhancement: Dynamic exit replacement . . . . .	65
4.6.1 The issue . . . . .	65
4.6.2 How this is addressed in z/OS 1.12 . . . . .	65
4.6.3 Restrictions or considerations . . . . .	66
4.6.4 Rollback . . . . .	66
4.6.5 Further information . . . . .	66

4.7	SETPROG enhancement: SVC enhancement	66
4.7.1	The issue	67
4.7.2	How this is addressed in z/OS 1.12	67
4.7.3	Restrictions or considerations	67
4.7.4	Rollback	67
4.7.5	Further information	67
4.8	Enhancements to LLA processing	67
4.8.1	The issue	67
4.8.2	How this is addressed in z/OS 1.12	68
4.8.3	Restrictions or considerations	68
4.8.4	Rollback	68
4.8.5	Further information	68
4.9	System REXX availability enhancements	68
4.9.1	The issue	68
4.9.2	How this is addressed in z/OS 1.9 and later	69
4.9.3	Restrictions or considerations	70
4.9.4	Rollback	70
4.9.5	Further information	70
4.10	Dynamically change system symbols	70
4.10.1	The issue	71
4.10.2	How this is addressed in z/OS 2.1	71
4.10.3	IEASYMU2 and IEASYMUP usage	72
4.10.4	Restrictions or considerations	72
4.10.5	Rollback	72
4.10.6	Further information	72
4.11	Add more than RESERVED number of CPs without an IPL	73
4.11.1	The issue	73
4.11.2	How this is addressed in z/OS 1.10	73
4.11.3	How this is addressed in z/OS 2.1	73
4.11.4	Restrictions or considerations	74
4.11.5	Rollback	75
4.11.6	Further information	75
4.12	Switching between primary and secondary DASD without an IPL	75
4.12.1	The issue	75
4.12.2	How this is addressed in z/OS 1.10	75
4.12.3	Restrictions or considerations	76
4.12.4	Rollback	76
4.12.5	Further information	76
4.13	Specify NOBUFFS action (SMF) at the log stream level	76
4.13.1	The issue	76
4.13.2	How this is addressed in z/OS 1.12	76
4.13.3	Restrictions or considerations	77
4.13.4	Rollback	77
4.13.5	Further information	77
4.14	Forcing hung commands to avoid an IPL	77
4.14.1	The issue	77
4.14.2	How this is addressed in z/OS 1.13	77
4.14.3	Restrictions or considerations	78
4.14.4	Rollback	78
4.14.5	Further information	78
4.15	Dynamically configure storage-class memory (SCM)	78
4.15.1	The issue	78
4.15.2	How this is addressed in z/OS 1.13	79

4.15.3	Restrictions or considerations . . . . .	79
4.15.4	Rollback . . . . .	79
4.15.5	Further information . . . . .	79
4.16	End a task with new operand on FORCE . . . . .	79
4.16.1	The issue . . . . .	79
4.16.2	How this is addressed in z/OS 2.1 . . . . .	79
4.16.3	Restrictions or considerations . . . . .	80
4.16.4	Rollback . . . . .	80
4.16.5	Further information . . . . .	80
4.17	Auto-reply for WTOR . . . . .	80
4.17.1	The issue . . . . .	80
4.17.2	How this is addressed in z/OS 1.12 . . . . .	80
4.17.3	Restrictions or considerations . . . . .	81
4.17.4	Rollback . . . . .	81
4.17.5	Further information . . . . .	81
4.18	DCCF support for WTOR Auto-Reply . . . . .	81
4.18.1	The issue . . . . .	81
4.18.2	How this is addressed in z/OS 2.1 . . . . .	82
4.18.3	Restrictions or considerations . . . . .	82
4.18.4	Rollback . . . . .	82
4.18.5	Further information . . . . .	82
4.19	New MODIFY VLF command . . . . .	83
4.19.1	The issue . . . . .	83
4.19.2	How this is addressed in z/OS 2.1 . . . . .	83
4.19.3	Restrictions or considerations . . . . .	83
4.19.4	Rollback . . . . .	83
4.19.5	Further information . . . . .	83
4.20	Add/Remove MCS consoles dynamically . . . . .	84
4.20.1	The issue . . . . .	84
4.20.2	How this is addressed in z/OS 2.1 . . . . .	84
4.20.3	How this is addressed in z/OS 1.10 . . . . .	85
4.20.4	Restrictions or considerations . . . . .	85
4.20.5	Rollback . . . . .	85
4.20.6	Further information . . . . .	85
4.21	New Message Flood Automation . . . . .	85
4.21.1	The issue . . . . .	85
4.21.2	How this is addressed in z/OS 1.9 . . . . .	85
4.21.3	How this is addressed in z/OS 1.11 . . . . .	86
4.21.4	How this is addressed in z/OS 1.13 . . . . .	86
4.21.5	Restrictions or considerations . . . . .	86
4.21.6	Rollback . . . . .	87
4.21.7	Further information . . . . .	87
4.22	Added LPA defer wait capability . . . . .	87
4.22.1	The issue . . . . .	87
4.22.2	How this is addressed in z/OS 1.12 . . . . .	87
4.22.3	Restrictions or considerations . . . . .	87
4.22.4	Rollback . . . . .	87
4.22.5	Further information . . . . .	88
4.23	Auto-IPL using DIAGxx . . . . .	88
4.23.1	The issue . . . . .	88
4.23.2	How this is addressed in z/OS 1.10 . . . . .	88
4.23.3	Restrictions or considerations . . . . .	88
4.23.4	Rollback . . . . .	89



4.23.5 Further information . . . . .	89
4.24 z/OS reusable address space support and usage . . . . .	89
4.24.1 The issue . . . . .	89
4.24.2 How this is addressed since z/OS 1.9. . . . .	90
4.24.3 Restrictions or considerations. . . . .	91
4.24.4 Rollback . . . . .	92
4.24.5 Further information . . . . .	92
4.25 z/OS report on linkage indexes. . . . .	93
4.25.1 The issue . . . . .	93
4.25.2 How this is addressed in z/OS 1.9 . . . . .	93
4.25.3 Restrictions or considerations. . . . .	93
4.25.4 Rollback . . . . .	93
4.25.5 Further information . . . . .	94
<b>Chapter 5. z/OS enhancements: GRS . . . . .</b>	<b>95</b>
5.1 Moving GRS contention notification system role. . . . .	96
5.1.1 The issue . . . . .	96
5.1.2 How this is addressed in z/OS 1.8 . . . . .	96
5.1.3 Restrictions or considerations. . . . .	97
5.1.4 Rollback . . . . .	97
5.1.5 Further information . . . . .	97
5.2 Control the maximum number of ENQs in a system . . . . .	97
5.2.1 The issue . . . . .	97
5.2.2 How this is addressed in z/OS 1.8 . . . . .	97
5.2.3 Restrictions or considerations. . . . .	97
5.2.4 Rollback . . . . .	97
5.2.5 Further information . . . . .	97
5.3 Changing to GRS without sysplex IPL . . . . .	98
5.3.1 The issue . . . . .	98
5.3.2 How this is addressed in z/OS 1.10 . . . . .	98
5.3.3 Restrictions or considerations. . . . .	98
5.3.4 Rollback . . . . .	98
5.3.5 Further information . . . . .	98
<b>Chapter 6. z/OS enhancements: Resource Recovery Services . . . . .</b>	<b>99</b>
6.1 RRS internal restart. . . . .	100
6.1.1 The issue . . . . .	100
6.1.2 How this is addressed in z/OS 2.1 . . . . .	100
6.1.3 Restrictions or considerations. . . . .	101
6.1.4 Rollback . . . . .	101
6.1.5 Further information . . . . .	101
6.2 Clean RRS shutdown option. . . . .	101
6.2.1 The issue . . . . .	101
6.2.2 How this is addressed in z/OS 1.13 . . . . .	102
6.2.3 Restrictions or considerations. . . . .	102
6.2.4 Rollback . . . . .	102
6.2.5 Further information . . . . .	102
6.3 Ability to dynamically turn on and off RRS archive log . . . . .	102
6.3.1 The issue . . . . .	102
6.3.2 How this is addressed in z/OS 1.10 . . . . .	103
6.3.3 Restrictions or considerations. . . . .	103
6.3.4 Rollback . . . . .	103
6.3.5 Further information . . . . .	103

<b>Chapter 7. z/OS enhancements: XCF and System Logger</b>	<b>105</b>
7.1 System Status Detection Partitioning Protocol	106
7.1.1 The issue	106
7.1.2 How this is addressed in z/OS 1.11	106
7.1.3 Restrictions or considerations	107
7.1.4 Rollback	107
7.1.5 Further information	107
7.2 CRITICALPAGING function for HyperSwap environments	107
7.2.1 The issue	107
7.2.2 How this is addressed in z/OS 1.12	108
7.2.3 Restrictions or considerations	108
7.2.4 Rollback	108
7.2.5 Further information	108
7.3 Non-disruptive CF dump	108
7.3.1 The issue	109
7.3.2 How this is addressed in z/OS 1.13	109
7.3.3 Restrictions or considerations	109
7.3.4 Rollback	109
7.3.5 Further information	109
7.4 XCF command to get structures to primary CF	109
7.4.1 The issue	109
7.4.2 How this is addressed in z/OS 1.8	110
7.4.3 Restrictions or considerations	110
7.4.4 Rollback	110
7.4.5 Further information	110
7.5 Force disconnection or deletion of a log stream without restarting the Logger address space	110
7.5.1 The issue	111
7.5.2 How this is addressed in z/OS 1.11	111
7.5.3 Restrictions or considerations	111
7.5.4 Further information	111
7.6 IXGCNFxx PARM	111
7.6.1 The issue	111
7.6.2 How this is addressed in z/OS 1.13	112
7.6.3 Restrictions or considerations	112
7.6.4 Rollback	112
7.6.5 Further information	112
7.7 System Logger threshold messages	112
7.7.1 The issue	112
7.7.2 How this is addressed in z/OS 2.1	112
7.7.3 Restrictions or considerations	112
7.7.4 Rollback	113
7.7.5 Further information	113
7.8 Multiple logstream/data set management	113
7.8.1 The issue	113
7.8.2 How this is addressed in z/OS 1.11	113
7.8.3 Restrictions or considerations	113
7.8.4 Rollback	113
7.8.5 Further information	113
7.9 Dynamic log stream updates for duplexing	114
7.9.1 The issue	114
7.9.2 How this is addressed in z/OS 1.10	114
7.9.3 Restrictions or considerations	114

7.9.4 Rollback .....	114
7.9.5 Further information .....	114
<b>Chapter 8. z/OS enhancements: Communications Server .....</b>	<b>115</b>
8.1 Specify a hot standby server to Sysplex Distributor .....	116
8.1.1 The issue .....	116
8.1.2 How this is addressed in z/OS 1.12 .....	116
8.1.3 Restrictions or considerations .....	116
8.1.4 Rollback .....	117
8.1.5 Further information .....	117
8.2 Resolver start error tolerance .....	117
8.2.1 The issue .....	117
8.2.2 How this is addressed in z/OS 2.1 .....	117
8.2.3 Restrictions or considerations .....	118
8.2.4 Rollback .....	118
8.2.5 Further information .....	119
8.3 Command to verify TCP profile syntax .....	119
8.3.1 The issue .....	119
8.3.2 How this is addressed in z/OS 2.1 .....	119
8.3.3 Restrictions or considerations .....	119
8.3.4 Rollback .....	119
8.3.5 Further information .....	120
8.4 RPCBIND/NFS re-registration .....	120
8.4.1 The issue .....	120
8.4.2 How this is addressed in z/OS 2.1 .....	120
8.4.3 Restrictions or considerations .....	120
8.4.4 Rollback .....	120
8.4.5 Further information .....	120
<b>Chapter 9. z/OS enhancements: DFSMS .....</b>	<b>121</b>
9.1 Dynamically activate changes to DEVSUPxx member .....	122
9.1.1 The issue .....	122
9.1.2 How this is addressed in z/OS 1.8 .....	122
9.1.3 How this is addressed in z/OS 1.10 .....	122
9.1.4 Restrictions or considerations .....	122
9.1.5 Rollback .....	122
9.1.6 Further information .....	122
9.2 Dynamically modify size of SMSPDSE hiperspaces .....	123
9.2.1 The issue .....	123
9.2.2 How this is addressed in z/OS 1.8 .....	123
9.2.3 Restrictions or considerations .....	123
9.2.4 Rollback .....	123
9.2.5 Further information .....	123
9.3 VSAM CA reclaim for KSDSs .....	124
9.3.1 The issue .....	124
9.3.2 How this is addressed in z/OS 1.12 .....	124
9.3.3 Restrictions or considerations .....	125
9.3.4 Rollback .....	125
9.3.5 Further information .....	125
9.4 Support for very large catalogs and catalogs on extended address volumes (EAVs) .....	125
9.4.1 The issue .....	125
9.4.2 How this is addressed in z/OS 1.12 .....	126
9.4.3 Restrictions or considerations .....	126

9.4.4	Rollback .....	126
9.4.5	Further information .....	126
9.5	Minimizing application disruption during catalog maintenance .....	126
9.5.1	The issue .....	126
9.5.2	How this is addressed in z/OS 2.1 .....	127
9.5.3	Restrictions or considerations .....	127
9.5.4	Rollback .....	127
9.5.5	Further information .....	127
9.6	DADSM and CVAF support for dynamic corrective service activation .....	127
9.6.1	The issue .....	128
9.6.2	How this is addressed in z/OS 1.13 .....	128
9.6.3	Restrictions or considerations .....	128
9.6.4	Rollback .....	128
9.6.5	Further information .....	128
9.7	DFSMSHsm Control Data Set backup enhancements .....	128
9.7.1	The issue .....	128
9.7.2	How this is addressed in z/OS 1.13 .....	129
9.7.3	Restrictions or considerations .....	129
9.7.4	Rollback .....	129
9.7.5	Further information .....	129
9.8	Allow XRC primary volumes to be offline when the XSTART and XADDPAIR commands are issued .....	129
9.8.1	The issue .....	129
9.8.2	How this is addressed in z/OS 2.1 .....	130
9.8.3	Restrictions or considerations .....	130
9.8.4	Rollback .....	130
9.8.5	Further information .....	130
9.9	Metro Mirror (PPRC) support for Remote Pair FlashCopy .....	130
9.9.1	The issue .....	130
9.9.2	How this is addressed in z/OS 1.11 .....	131
9.9.3	Restrictions or considerations .....	132
9.9.4	Rollback .....	132
9.9.5	Further information .....	132
9.10	Remove need to offline/online volume after VTOC change .....	132
9.10.1	The issue .....	132
9.10.2	How this is addressed in z/OS 1.13 .....	132
9.10.3	Restrictions or considerations .....	133
9.10.4	Rollback .....	133
9.10.5	Further information .....	133
9.11	Dynamically update maximum number of tape drives OAM allocates .....	133
9.11.1	The issue .....	133
9.11.2	How this is addressed in z/OS 1.13 .....	134
9.11.3	Restrictions or considerations .....	134
9.11.4	Rollback .....	134
9.11.5	Further information .....	134
<b>Chapter 10.</b>	<b>z/OS enhancements: Distributed File Service (zFS)</b> .....	<b>135</b>
10.1	zFS internal restart .....	136
10.1.1	The issue .....	136
10.1.2	How this is addressed in z/OS 1.13 .....	136
10.1.3	Restrictions or considerations .....	137
10.1.4	Rollback .....	137
10.1.5	Further information .....	137

10.2 zFS automatic re-enable of disabled file systems . . . . .	137
10.2.1 The issue . . . . .	138
10.2.2 How this is addressed in z/OS 1.13 . . . . .	138
10.2.3 Restrictions or considerations . . . . .	139
10.2.4 Rollback . . . . .	139
10.2.5 Further information . . . . .	139
<b>Chapter 11. z/OS enhancements: Security Server (RACF) . . . . .</b>	<b>141</b>
11.1 Sample passphrase ICHPWX11 exit uses System REXX . . . . .	142
11.1.1 The issue . . . . .	142
11.1.2 How this is addressed in z/OS 1.9 . . . . .	142
11.1.3 Restrictions or considerations . . . . .	142
11.1.4 Rollback . . . . .	142
11.1.5 Further information . . . . .	142
11.2 Sample password ICHPWX01 exit uses System REXX . . . . .	142
<b>Chapter 12. z/OS enhancements: ICSF . . . . .</b>	<b>145</b>
12.1 ICSF: Improve product stability . . . . .	146
12.1.1 The issue . . . . .	146
12.1.2 How this is addressed in z/OS 1.12 . . . . .	146
12.1.3 Restrictions or considerations . . . . .	146
12.1.4 Rollback . . . . .	146
12.1.5 Further information . . . . .	147
<b>Chapter 13. z/OS enhancements: JES2 . . . . .</b>	<b>149</b>
13.1 Dynamic JES2 exit support . . . . .	150
13.1.1 The issue . . . . .	150
13.1.2 How this is addressed in z/OS 1.9 . . . . .	150
13.1.3 Restrictions or considerations . . . . .	150
13.1.4 Rollback . . . . .	151
13.1.5 Further information . . . . .	151
13.2 Dynamic changes to JES2 parameters . . . . .	151
13.2.1 The issue . . . . .	151
13.2.2 How this is addressed in z/OS 1.13 . . . . .	151
13.2.3 Restrictions or considerations . . . . .	151
13.2.4 Rollback . . . . .	151
13.2.5 Further information . . . . .	151
13.3 Early release of job log from long running jobs and started tasks . . . . .	152
13.3.1 The issue . . . . .	152
13.3.2 How this is addressed in z/OS 1.13 . . . . .	152
13.3.3 Restrictions or considerations . . . . .	152
13.3.4 Rollback . . . . .	152
13.3.5 Further information . . . . .	152
13.4 Use SPIN=UNALLOC JCL keyword to spin off spool files . . . . .	153
13.4.1 How this is addressed in z/OS 1.13 . . . . .	153
13.4.2 How this is addressed in z/OS 1.13 . . . . .	153
13.4.3 Restrictions or considerations . . . . .	153
13.4.4 Rollback . . . . .	153
13.4.5 Further information . . . . .	154
13.5 Ability to stop a JES2 job at the end of the current step . . . . .	154
13.5.1 The issue . . . . .	154
13.5.2 How this is addressed in z/OS 1.13 . . . . .	154
13.5.3 Restrictions or considerations . . . . .	154
13.5.4 Rollback . . . . .	154

13.5.5 Further information . . . . .	154
13.6 Dynamically modify a JES2 spool volume . . . . .	154
13.6.1 The issue . . . . .	155
13.6.2 How this is addressed in z/OS 1.13 . . . . .	155
13.6.3 Restrictions or considerations . . . . .	155
13.6.4 Rollback . . . . .	155
13.6.5 Further information . . . . .	155
13.7 Support for over four billion spin data sets . . . . .	155
13.7.1 The issue . . . . .	156
13.7.2 How this is addressed in z/OS 2.1 . . . . .	156
13.7.3 Restrictions or considerations . . . . .	156
13.7.4 Rollback . . . . .	156
13.7.5 Further information . . . . .	156
<b>Chapter 14. z/OS enhancements: JES3 . . . . .</b>	<b>157</b>
14.1 Dynamically modify TCP/IP NJE information . . . . .	158
14.1.1 The issue . . . . .	158
14.1.2 How this is addressed in z/OS 1.8 . . . . .	158
14.1.3 Restrictions or considerations . . . . .	159
14.1.4 Rollback . . . . .	159
14.1.5 Further information . . . . .	159
14.2 Ability to dynamically add spool volumes to JES3 . . . . .	159
14.2.1 The issue . . . . .	159
14.2.2 How this is addressed in z/OS 1.13 . . . . .	160
14.2.3 Restrictions or considerations . . . . .	161
14.2.4 Rollback . . . . .	161
14.2.5 Further information . . . . .	161
14.3 Ability to dynamically remove spool volumes from JES3 . . . . .	161
14.3.1 The issue . . . . .	162
14.3.2 How this is addressed in z/OS 2.1 . . . . .	162
14.3.3 Restrictions or considerations . . . . .	164
14.3.4 Rollback . . . . .	164
14.3.5 Further information . . . . .	164
<b>Chapter 15. z/OS enhancements: Infoprint Server. . . . .</b>	<b>165</b>
15.1 Dynamic configuration changes for most InfoPrint options . . . . .	166
15.1.1 The issue . . . . .	166
15.1.2 How this is addressed in z/OS 2.1 . . . . .	166
15.1.3 Restrictions or considerations . . . . .	167
15.1.4 Rollback . . . . .	167
15.1.5 Further information . . . . .	167
15.2 Print Services Facility (PSF) installation is disassociated from SYS1.NUCLEUS. . .	167
15.2.1 The issue . . . . .	168
15.2.2 How this is addressed in z/OS V2.1 . . . . .	168
15.2.3 Restrictions or considerations . . . . .	168
15.2.4 Rollback . . . . .	168
15.2.5 Further information . . . . .	168
<b>Chapter 16. z/OS enhancements: TSO/E . . . . .</b>	<b>169</b>
16.1 Support for VTAM unconditional reconnect for TSO/E users . . . . .	170
16.1.1 The issue . . . . .	170
16.1.2 How this is addressed in z/OS 1.11 . . . . .	170
16.1.3 Restrictions or considerations . . . . .	170
16.1.4 Rollback . . . . .	171

16.1.5 Further information .....	171
<b>Chapter 17. z/OS enhancements: z/OS UNIX</b> .....	173
17.1 Support dynamic modification of AUTOCDT setting .....	174
17.1.1 The issue .....	174
17.1.2 How this is addressed in z/OS V1.13 .....	174
17.1.3 Restrictions or considerations .....	174
17.1.4 Further information .....	174
17.2 Prevent content overlay during mount .....	174
17.2.1 The issue .....	175
17.2.2 How this is addressed in z/OS 1.13 .....	175
17.2.3 Restrictions or considerations .....	177
17.2.4 Rollback .....	177
17.2.5 Further information .....	177
17.3 Change sysplex root data set without sysplex IPL .....	177
17.3.1 The issue .....	177
17.3.2 How this is addressed in z/OS 1.10 .....	178
17.3.3 Restrictions or considerations .....	178
17.3.4 Rollback .....	178
17.3.5 Further information .....	178
17.4 Support for alternate sysplex root data set .....	178
17.4.1 The issue .....	178
17.4.2 How this is addressed in z/OS 1.10 .....	178
17.4.3 Restrictions or considerations .....	179
17.4.4 Rollback .....	179
17.4.5 Further information .....	179
17.5 Change number of common inet ports without OMVS restart .....	179
17.5.1 The issue .....	179
17.5.2 How this is addressed in z/OS 1.12 .....	179
17.5.3 Restrictions or considerations .....	179
17.5.4 Rollback .....	180
17.5.5 Further information .....	180
17.6 Remount with same mount mode .....	180
17.6.1 The issue .....	180
17.6.2 How this is addressed in z/OS 1.11 .....	180
17.6.3 Restrictions or considerations .....	180
17.6.4 Rollback .....	180
17.6.5 Further information .....	180
<b>Part 2. z/OS middleware</b> .....	181
<b>Chapter 18. CICS considerations</b> .....	183
18.1 Possible reasons to schedule stopping a CICS region .....	184
18.2 Techniques to reduce unavailability through configuration changes .....	184
18.2.1 Reloading programs .....	184
18.2.2 Partitioned Data Set Load Library versus Extended .....	185
18.2.3 Dynamic program library management .....	186
18.2.4 Dynamic allocation of files and Extrapartition Transient Data Queues .....	186
18.2.5 Changes to CICS System Initialization Table values .....	187
18.2.6 Reload an updated RACF Certificate key ring .....	189
18.2.7 Change of Local Time on z/OS system .....	189
18.2.8 RACF changes .....	191
18.2.9 Tracking dynamic changes to a CICS region .....	192
18.2.10 Applying Service .....	192

18.2.11 Upgrading CICS .....	195
18.3 Reduction of preventive recycles of CICS .....	197
18.3.1 Storage shortage .....	197
18.4 Reducing shutdowns through runtime dependencies .....	207
18.5 Reducing shutdowns to allow resource access by non-CICS tasks .....	209
18.5.1 Backups .....	209
18.5.2 Batch access to data sets .....	210
18.6 CICS still must be stopped, so what next? .....	214
<b>Chapter 19. DB2 considerations</b> .....	215
19.1 DB2 subsystem or data sharing group member outage .....	216
19.1.1 Applying DB2 service (PTF/RSU/CST) .....	216
19.1.2 Version to version upgrade/fallback .....	226
19.1.3 DSNZPARM changes .....	227
19.1.4 Convert BSDS to Extended RBA/LRSN in DB2 11 .....	227
19.1.5 Dynamic addition of active logs .....	228
19.2 Database changes .....	228
19.2.1 DB2 for z/OS Version 8 .....	229
19.2.2 DB2 for z/OS 9 .....	229
19.2.3 DB2 for z/OS 10 .....	233
19.2.4 DB2 for z/OS 11 .....	242
<b>Chapter 20. IMS considerations</b> .....	251
20.1 IMS database considerations .....	252
20.1.1 Database reorganization .....	252
20.1.2 Dynamic database buffering .....	254
20.1.3 Miscellaneous database considerations .....	255
20.2 IMS Transaction Manager Considerations .....	256
20.2.1 Dynamic change for OTMA routing descriptors .....	256
20.2.2 Dynamically change the IMS Connect configuration .....	256
20.2.3 MSC dynamic reconfiguration .....	257
20.3 IMS system considerations .....	257
20.3.1 Member Online Change (MOLC) .....	258
20.3.2 ACBLIB dynamic allocation .....	258
20.3.3 ACBIN64 .....	258
20.3.4 Language Environment dynamic runtime options .....	259
20.3.5 Refreshable user exits .....	259
20.3.6 Dynamic LOCKTIME .....	259
20.3.7 Dynamic resource definition (DRD) .....	260
20.3.8 DBRC .....	260
20.3.9 IMS Version upgrades without an IPL .....	261
20.4 Resources still requiring a recycle of IMS to change .....	261
20.4.1 Variable storage pools .....	261
20.4.2 Fixed size (SPM) storage pools .....	262
20.4.3 Log buffers .....	262
<b>Chapter 21. WebSphere MQ for z/OS</b> .....	263
21.1 Basic concepts and terms .....	264
21.1.1 Messages / message queues .....	264
21.1.2 Queues .....	264
21.1.3 Unit of work .....	264
21.2 Availability and planned outages .....	265
21.2.1 Minimizing a queue manager outage due to an image IPL .....	265
21.2.2 Availability of messaging in a QSG during an image IPL .....	265



21.2.3	Can a WebSphere MQ channel be automatically rerouted if a queue manager is made unavailable? . . . . .	266
21.2.4	Can a queue remain available if a queue manager is stopped and you are not using shared queues? . . . . .	266
21.2.5	How can a message be processed by a different queue manager or queue without changing an application? . . . . .	267
21.2.6	Improving availability to client applications during a planned outages . . . . .	268
21.2.7	Further information . . . . .	268
21.3	Applying maintenance during subsystem operation . . . . .	269
21.3.1	Concurrent code levels . . . . .	269
21.3.2	Do you need to bind DB2 application plans following maintenance? . . . . .	269
21.3.3	Catering to a DB2 outage in a QSG . . . . .	271
21.3.4	Performing maintenance on a queue manager in a cluster . . . . .	272
21.3.5	How can a queue manager be made to slow the rate of messages in the event of an IMS message flood? . . . . .	272
21.4	Changes that can be made dynamically . . . . .	272
21.4.1	Can the buffer pools be changed while the queue manager is up? . . . . .	273
21.5	Changes requiring a queue manager restart or IPL . . . . .	273
21.5.1	Does an early code upgrade need a queue manager restart or IPL? . . . . .	273
21.6	Error recovery (without a system or subsystem restart) . . . . .	274
21.6.1	What happens to coordinated unit of recovery in a QSG? . . . . .	274
21.6.2	WebSphere MQ peer recovery in a QSG . . . . .	274
21.6.3	Toleration for CF connectivity loss and recovery enhancements . . . . .	275
<b>Appendix A. Items requiring an IPL or recycle. . . . .</b>		<b>277</b>
Items requiring an IPL or recycle to implement . . . . .		278
<b>Appendix B. DB2 UNIX file system maintenance considerations . . . . .</b>		<b>283</b>
Type 2 and type 4 connectivity. . . . .		284
Example of DB2 maintenance and the zFS . . . . .		284
Allocation scripts . . . . .		287
Executions of the scripts . . . . .		291
Summary. . . . .		296
<b>Related publications . . . . .</b>		<b>297</b>
IBM Redbooks . . . . .		297
Other publications . . . . .		297
Online resources . . . . .		298
Help from IBM . . . . .		298



# Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

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

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

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


## COPYRIGHT LICENSE:

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

# Trademarks

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

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

CICS Explorer®	IBM®	System z10®
CICSplex®	IMS™	System z®
CICS®	Language Environment®	Tivoli®
DB2®	MVS™	VTAM®
FlashCopy®	NetView®	WebSphere®
GDPS®	OMEGAMON®	z/OS®
Geographically Dispersed Parallel Sysplex™	Parallel Sysplex®	z/VM®
Global Technology Services®	PrintWay™	z10™
HyperSwap®	RACF®	z9®
IA®	Redbooks®	
	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

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

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

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

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

## Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

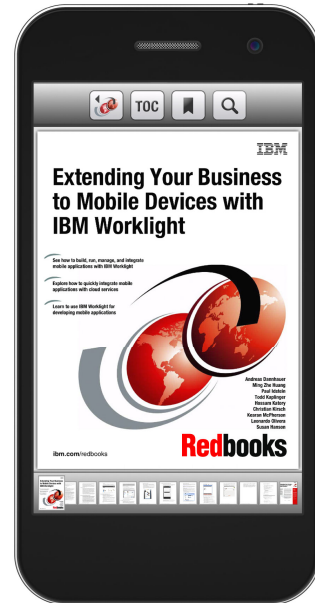
Get the latest version of the Redbooks Mobile App



iOS

**Download  
Now**

Android



## Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



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

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

# Preface

This IBM® Redbooks® publication is intended to make System Programmers, Operators, and Availability Managers aware of the enhancements to recent releases of IBM z/OS® and its major subsystems in the area of planned outage avoidance. It is a follow-on to, rather than a replacement for, *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

Its primary objective is to bring together in one place information that is already available, but widely dispersed. It also presents a different perspective on planned outage avoidance. Most businesses care about application availability rather than the availability of a specific system. Also, a planned outage is not necessarily a bad thing, if it does not affect application availability. In fact, running for too long without an IPL or subsystem restart might have a negative impact on application availability because it impacts your ability to apply preventive service. Therefore, this book places more focus on decoupling the ability to make changes and updates to your system from IPLing or restarting your systems.

## Authors

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

**Frank Kyne** is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of IBM Parallel Sysplex® and High Availability. Before joining the ITSO 15 years ago, Frank worked in IBM Ireland as an MVS™ System Programmer.

**Andy Clifton** is a IBM CICS® Tester based at IBM Hursley in England who holds a degree in Mathematics and Computing from Bath University. Having 26 years of experience testing CICS, Andy has worked on many releases of CICS, including products such as CICS/VM, CICS/VSE, CICS/ESA, and CICS Transaction Server. Andy has previously worked on IBM Redbooks publications related to Parallel Sysplex and the Parallel Sysplex Trainer. He has been involved in several IBM cross-lab projects in Toronto, Poughkeepsie, and San Jose as a CICS expert. Andy is currently in the CICS Integration Test Team at Hursley and is the CICS expert in the IBM Consolidated Service Test.

**Justin Deane** currently works as a software support engineer for IBM WebSphere® MQ for z/OS based in Hursley, UK. Plucked from a world of mathematics and particle physics research in Durham University, England to work for IBM UK, he has now gained 14 years of support experience for messaging middleware on z/OS. He was the first and initially the only Level 3 support representative for message broker on the mainframe. He is also a co-author of a former ITSO workshop, *WebSphere MQ Integrator Administration on z/OS*.

**Fernando Ferreira** is an IT Specialist working for the IBM Advanced Technical Skills (ATS) Latin America group in IBM Brazil. Fernando joined IBM in 1996 as a Customer Technical Specialist, supporting customers in the finance industry. He now supports Customer Technical Specialists in complex or strategic opportunities. Before joining IBM, he worked for nine years for an IBM customer. Fernando is a subject matter expert on WebSphere Application Server for z/OS, a member of the IBM zChampions team, and a member of the IBM Academy of Technology.

**Richard Gunjal** is an IT Architect in IBM Global Technology Services® GD in IBM India. He has 16 years of IT experience in various areas, including systems programming. He holds a master degree in Information Management Computer Science from University of Wales and joined IBM Germany in 2005. His areas of expertise include IBM System z®, z/OS, DB2®, and availability management.

**Chuck Laurent** is a Director, System Programming, in a large financial institution in the United States. He is an IBM Certified System Programmer with 30+ years of experience in mainframe system programming and technical support. His areas of expertise include installation and support of many z/OS and ISV products.

**John Papp** is a z/OS Systems Programmer for IBM Global Services in the United States. He has many years of experience in IT. He started in Computer Operations, then quickly moved to handle all aspects of VM and z/OS Systems Programming. His current area of specialization is in z/OS UNIX System Services. Before joining IBM, John worked for State Street Bank.

**Judy Ruby-Brown** is an Executive I/T Specialist in the United States. She has 25 years of experience in DB2 for z/OS technical support in IBM Advanced Technical Sales Support. She holds a degree in mathematics from the University of Oklahoma. Her areas of expertise include IBM Parallel Sysplex and DB2 data sharing, including virtual storage, recovery, high availability, and off-site disaster recovery. She has performed DB2 data sharing consulting and Parallel Sysplex Infrastructure Health Checks services for over 25 of IBM's largest clients. She has reviewed many IBM Redbooks documents and co-authored *DB2 for z/OS: Data Sharing in a Nutshell*, SG24-7322, *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370, and *DB2 for z/OS and OS/390: Ready for Java*, SG24-6435. She regularly presents on disaster recovery and Parallel Sysplex topics at user group conferences such as IDUG, IOD, IBM DB2 Technical Conference, and SHARE. She has authored multiple Washington System Center flashes and white papers.

**Maida Snapper** is an IBM IMS™ specialist with the IBM Silicon Valley Lab. Maida holds a degree in Mathematics from the City University of New York and has been working with IMS for over 30 years in various roles including application development, systems programming, and database administration. Maida is based in New York and provides technical support, services, and training for IBM IMS customers worldwide.

**Michael Stephen** is a Senior Software Engineer at IBM Poughkeepsie. He is currently Team Lead for Level 2 Support for WebSphere Application Server z/OS and has been in WebSphere Application Server Level 2 support since its inception in 1999. Before his current group, Mike spent 12 years in the z/OS Level 2 Support area. Mike has presented to customers about the new releases of WebSphere Application Server on z/OS, and has presented various WebSphere and support topics at SHARE User conferences.

**Dave Viguers** is retired from IBM but continues to work as a supplemental employee working from his home in Texas. He has 43 years of experience in IMS. His primary area of expertise is IMS performance and sysplex implementation. He has participated in the writing of many IBM Redbooks publications and teaches IMS sysplex implementation classes.

**Marna Walle** is a Senior Technical Staff Member at IBM Poughkeepsie. She has worked in the z/OS Development organization for 24 years in the area of system installation and migration. She holds a bachelor's degree in Computer Science from the University of Florida and a master's degree in Computer Science from New York University. She is a frequent speaker around the world on z/OS migration and using new functions, helping numerous customers upgrade to new releases of z/OS. Marna has also written extensively for technical publications on z/OS system programmer topics.



**Yvonne Zemotel** was a Senior IT Specialist with IBM US. She had many years of experience in mainframe application, systems programming, technical support, and project management. Her areas of expertise included installation, maintenance, and support of z/OS and ISV products, and project management for mainframe initiatives. She was part of the team that developed the original *z/OS Planned Outage Avoidance Checklist*, SG24-7328. Before joining IBM, she worked at State Street Bank for over 17 years. Sadly, Yvonne passed away before this book could be finalized. Her contributions to the team and to the publication were greatly appreciated.

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

Richard Conway  
Mike Ebbers  
Bob Haimowitz

**International Technical Support Organization, Poughkeepsie Center**

John Campbell  
**IBM UK**

Paul Ayer  
Charles Bryant  
John Cowel  
Jerry Dearing  
Stan Gildea  
Jack Hwang  
Scott Marcotte  
Jim McGillicuddy  
Jim Mulder  
Peter Relson  
Haaken Roberts  
Daniel Rosa  
Rodney Ross  
Anita Snell  
Paul Streitman  
**IBM US**

Thanks to the authors of the previous editions of this book.

- Authors of the first book in this series, *z/OS Planned Outage Avoidance Checklist*, SG24-7328, published in August 2006, were:

Brad Habbershaw  
Vivian Roberts  
Simon Truby  
Yvonne Zemotel

## Now you can become a published author, too!

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

Find out more about the residency program, browse the residency index, and apply online at:  
[ibm.com/redbooks/residencies.html](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 publications 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 email 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

## Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

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

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

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

- Stay current on recent Redbooks publications with RSS Feeds:

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



# Planned outage considerations

This chapter describes the concept of planned outages and describes how to have planned outages has changed over the last 40 years. It positions this book alongside an earlier IBM Redbooks publication on this topic, *z/OS Planned Outage Avoidance Checklist*, SG24-7328. It describes the layout of the remainder of this book.

This chapter includes the following sections:

- ▶ What is a planned outage?
- ▶ System availability or application availability?
- ▶ Planned outage avoidance or control?
- ▶ Differences between planned and unplanned outages
- ▶ Reasons for planned outages
- ▶ What this book covers
- ▶ Layout of this book
- ▶ Live Guest Relocation

## 1.1 What is a planned outage?

There was a time when all the users of mainframe systems were in the same building as the mainframe. As a result, all users of the system could easily be given advance notice of any upcoming outages of the system. Even when networks were implemented and the user population was spread over multiple locations, because they were all in the same company, they could be informed in advance of any planned interruptions to the service.

In those days, a planned outage truly was planned because not only did the IT department know about upcoming outages, so did all the users of the system, meaning that everyone could *plan* their work around the outage.

However, as time passed, the number of applications on the system and the number of users using those applications increased. This made it increasingly difficult to communicate with every user of the system to inform them about pending outages. This was the start of the dilution of the concept of planned outages. The IT department and some of the users knew about the outage in advance, but some users did not. To those that did not, the system was simply unavailable. To them, the fact that the IT department knew in advance that the system would be down was irrelevant.

Today, it is likely that the bulk of your users are not employees of your company. They access your data and applications through a multi-layered application architecture, often with no awareness that the applications they are using are communicating with your systems in the background. For those users, there is certainly no concept of planned outages; their application either works or it doesn't. And if it doesn't work, there is a good chance that your company will be on the evening news. Some companies now find that they learn about outages to their applications faster by monitoring Twitter than by relying on their own system management tools.

Another consideration for anyone responsible for maintaining a system or subsystem is the many aspects of globalization. An obvious one is the fact that people are traveling much more, so your customers that used to be local are now trying to access your system from anywhere in the world. So the middle of their day is now in the middle of your "planned outage" window.

Even if the users on each of your systems are in the same time zone as the system they are using, your data center in England might be running systems for subsidiaries of your company spread all over the world, in every time zone, and with different local and national holidays. The chances of finding a day or time when none of your users want to access your systems is often nearly zero.

The net effect of all of these changes is that the day of the planned outage is just about over. See Figure 1-1 on page 3. For people who are charged with maintaining the systems, subsystems, and applications, this can appear to be a daunting challenge. How can you serve your two masters:

- ▶ Users who demand 24x7 application availability (implying that your systems can never be taken down).
- ▶ Managers who demand 24x7 application availability (meaning that your systems must be taken down so that they can be kept up-to-date to minimize the risk of an unplanned outage).

The objective of this book is to help you keep both of these parties happy.

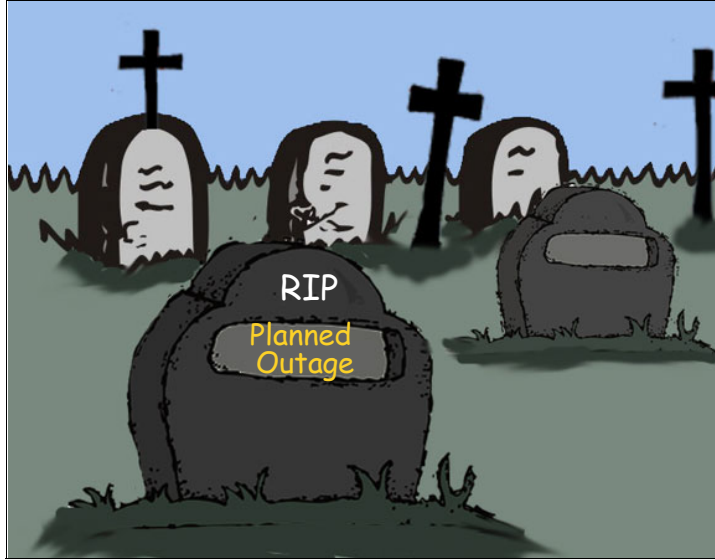


Figure 1-1 Future of planned outage concept

## 1.2 System availability or application availability?

Back in the early days of IT, when mainframes started to become prevalent, software was free, but hardware was expensive. As a result, each mainframe had to be shared between multiple applications because few applications could justify the expense of a mainframe of their own. As mainframes became larger and less expensive, and more applications wanted to share data between each other, it was natural that those applications would all exist within the same system.

As a result, it became the norm to associate an application with a system. If ApplicationA ran on system MVSA, and MVSA was down, then ApplicationA was down.

This paradigm continued until the introduction of Parallel Sysplex. Parallel Sysplex allows data to be shared, with integrity and minimal processor burden, between multiple operating system images. This meant that applications could now be run on multiple operating system images. This capability breaks the one-to-one relationship between system availability and application availability.

Today, most users of your mainframe applications might not even be aware that they are using a mainframe, much less *which* system they are using. They just want their application to be available. If their application runs on three systems, and two of them are down, the users neither know nor care. If their application is still working on the third system, they are happy.

This is good news for the staff who are responsible for maintaining the systems and subsystems that are being used by the applications. You can now shut down a system without impacting the availability of the applications running on that system<sup>1</sup>, thus helping to keep your users happy. Because you are able to schedule these outages to apply critical preventive maintenance, providing better availability in the future, your management should be happy.

---

<sup>1</sup> This assumes that the applications are using data sharing and dynamic workload routing.

One customer related an interesting experience. Because their systems supported online shopping (but not Parallel Sysplex), it was nearly impossible for them to schedule an outage to apply preventive service. As a result, they suffered an unacceptably high number of unplanned outages. In an attempt to protect their applications from those unplanned outages, they convinced the company to use data sharing and dynamic workload routing. A side effect of this new capability was that they were able to schedule more planned outages to apply preventive service and keep their systems more up-to-date. So they now had protection from unplanned outages and the ability to have more frequent planned outages. The interesting thing was that because they were able to be more proactive about applying service more frequently, the number of unplanned outages dropped to nearly zero.

There are a number of lessons here:

- ▶ It is vital to separate the idea of application availability from the action of taking a system or subsystem down. One no longer necessarily equals the other.
- ▶ As planned and unplanned outages blur into one in the eyes of your users, it is critical that you use whatever tools are available to maintain application availability across both planned and unplanned outages.
- ▶ There will always be situations that require a planned outage (these are described in 1.5, “Reasons for planned outages” on page 5), so you need some way to accommodate that reality while also protecting application availability.

In the z/OS world, the only way to decouple system availability from application availability is to fully embrace and use Parallel Sysplex data sharing *and* dynamic workload routing. It is not objective of this book to describe Parallel Sysplex. The following documents are helpful if you are interested in that topic:

- ▶ *Achieving the Highest Levels of Parallel Sysplex Availability*, SG24-6061
- ▶ *Parallel Sysplex Application Considerations*, SG24-6523
- ▶ *System z Parallel Sysplex Best Practices*, SG24-7817
- ▶ *DB2 for z/OS: Data Sharing in a Nutshell*, SG24-7322

## 1.3 Planned outage avoidance or control?

The predecessor to this book is *z/OS Planned Outage Avoidance Checklist*, SG24-7328. That document describes enhancements to z/OS (up to and including z/OS 1.7) that enable you to make changes to the system dynamically. Before those enhancements, making a particular change might have required an outage to the system or subsystem. This book continues in a similar vein (starting with z/OS 1.8 and going to z/OS 2.1). However, the focus is slightly different. Instead of talking about planned outage avoidance, the focus of this book is on decoupling system or subsystem changes from system or subsystem restarts.

Talking about planned outage *avoidance* implies that planned outages are something that can and should be avoided. However, that is neither fair nor accurate. Depending on your environment, applications, software mix, and availability requirements, the frequency with which you restart systems or subsystems will vary. The important thing is that these restarts should not affect your application availability. This book will help you understand these questions:

- ▶ What capabilities are available to allow you to make changes to your systems and subsystems without requiring an outage?
- ▶ What is the appropriate restart frequency for you?

The first of these objectives is based mainly on enhancements to z/OS and its major subsystems (IMS, CICS, DB2, and so on). This book also helps you to identify the appropriate restart frequency.

## 1.4 Differences between planned and unplanned outages

If you ask an applications user about the difference between a planned and an unplanned outage, they will probably be of the opinion that there is no difference. If they do not know in advance that the applications will be unavailable, then an outage is an outage.

However, from the perspective of the IT support staff, there is a huge difference. Planned outages can be prepared for and actions taken to try to minimize the impact. The outage is generally scheduled for a time that is intended to cause the least disruption. And, perhaps most important, everything is shut down in an orderly manner, meaning that it comes back up in a timely manner.

Unplanned outages are unpredictable. The outage can be caused by a hardware, software, network, or environmental problem. There is usually no ability to shut down applications in an orderly manner. And, they tend to happen at the worst possible time. Because of their unpredictable nature, it is difficult to plan for recovery from them, meaning that the duration of the outage is unpredictable.

Given a choice between the two, it is obviously preferable to have a planned outage rather than an unplanned one. When used wisely, planned outages might help you decrease the number of unplanned outages. For example, applying HIPER PTFs might help avoid an unplanned outage in the future.

The ideal situation is that your environment is configured in such a manner that all applications continue to be available even if one system or subsystem is down. This configuration provides the flexibility to schedule planned outages at an appropriate frequency, without adversely impacting application availability.

## 1.5 Reasons for planned outages

There are good and bad reasons for having a planned outage. Taken as a whole, good planned outages tend to improve application availability and minimize impact and disruption, whereas bad planned outages have the opposite effect.

The following are some good reasons for a planned outage:

- ▶ A disruptive CPC upgrade.

Most upgrades to an existing CPC can be performed non-disruptively. However, changing CPC types, from a z196 to a zEC12, for example, requires that all systems running on that CPC are stopped. This is an unavoidable outage.

- ▶ An upgrade to a new operating system or subsystem release or version.

Moving from one release of an operating system or subsystem to another always requires an IPL. This is an unavoidable outage.

- ▶ Mass application of many PTFs.

Although many PTFs can be applied concurrently, if you are applying hundreds of PTFs to a system, stop the system and IPL it from a new sysres that contains the to-be-applied PTFs.

- Applying critical PTFs that cannot be applied concurrently.

Some PTFs update parts of the system that can only be refreshed (or only refreshed safely) by running an IPL or subsystem restart. An example is a PTF that updates the MVS nucleus, or a module in LPA that cannot be safely replaced.

- IPL to address a problem with fragmented common storage, or common storage creep.

A proactive IPL can be good if it avoids a problem that would otherwise adversely affect the system in the future. However, base the IPL on the potential impact of the problem. For example, if common storage usage is growing at a rate that would consume all common storage in 12 weeks, the IPL frequency should be based on that information.

- IPL to “harden” dynamic changes

One of the common fears of installations that depend heavily on dynamic change capabilities is that they will forget to mirror some change back into the representative set of definitions (a SYS1.PARMLIB member, for example). Therefore, they schedule IPLs purely to ensure that all the changes are accurately reflected in the system definitions.

Strong system management practices and processes, combined with effective tools to compare the running system to the system definitions might give you the confidence to run for longer between such IPLs.

All of the above situations are either unavoidable, or else they are actions that are taken to avoid a possible service interruption in the future.

The following examples of some common *bad* reasons for a planned outage:

- Operator training.

It is important that operators are able to successfully run a system or subsystem restart, and handle any problems that might arise as a result of that. However, that is what test systems are for.

- Addressing problems that might not even exist any longer.

IPL at a specific frequency to address a known problem might be a good way to avoid potential future problems. However, still running an IPL at the same frequency five years after the initial problem probably is not good. More than likely, every piece of software in your inventory has been replaced with newer releases over that time. So maybe the problem no longer exists, and you are running an IPL to address a non-existent problem. Or maybe the problem is twice as bad now as it was then, in which case the IPL frequency should be adjusted.

- “Because we have always done it this way”

This is perhaps the most common, and certainly the most unacceptable, reason for a given planned outage policy.

Identifying the most appropriate restart frequency for your environment will be described in Chapter 2, “Project planning and user experiences” on page 9.

## 1.6 What this book covers

One of the objectives of this book is to inform you about enhancements to z/OS and its major subsystems that allow you to implement changes to that component without requiring a restart.

Another objective is to describe the ability to plan ahead so that future upgrades can be implemented dynamically.



This book also covers improvements in error recovery, so that the system or subsystems can now recover from problems that previously required a planned IPL or subsystem restart to resolve. One of the major enhancements in this arena is an extension of Parallel Sysplex called IBM GDPS® or Geographically Dispersed Parallel Sysplex™. This enhancement involves mainframes in different cities and different parts of the country. Part of GDPS is a function called IBM HyperSwap®. It is based on synchronous Peer-to-Peer Remote Copy (PPRC) technology for use within a single sysplex or data center. Data is copied from the primary storage device to an auxiliary storage device. During a failure of a system or storage device, recovery can occur with limited or no data loss automatically. GDPS is a large undertaking and there are other like-minded products, requiring the same amount of costs, planning, and labor that might limit data loss and protect integrity. For more information about GDPS, see *GDPS Family: An Introduction to Concepts and Capabilities*, SG24-6374.

Another type of enhancement described involves features that lengthen the time that a system or subsystem can run without requiring a restart. These enhancements are designed to give you added flexibility. Rather than being forced to IPL because you are running out of a resource, these enhancements allow the system to run for longer than most customers would want to run without a restart. This configuration eliminates them as an item that *forces* you to do a restart.

All of these changes are typically provided as part of the set of enhancements that are included with a new release or new version. However, sometimes enhancements of this type are shipped as a PTF. And sometimes the enhancements are included with one release, and then rolled back to a previous release with one or more PTFs. This book covers both situations.

**Note:** This book is targeted specifically at *planned* outage situations. There are numerous other IBM documents and IBM Redbooks publications that address high availability that are of interest if you are looking at *unplanned* outages. But that information is not repeated in this book.

Also, this book is intended to make you aware of enhancements and to provide a brief description of the enhancements and any considerations you should be aware of. For detailed implementation information, see the product publications for the product.

This book represents a point in time. Information about product enhancements is usually available from a number of publications or other sources of information. Each chapter in this book concludes with a list of information sources.

## 1.7 Layout of this book

In *z/OS Planned Outage Avoidance Checklist*, SG24-7328, the chapter contents are based on the typical roles and responsibilities of z/OS Systems Programmers seen in most installations from a z/OS viewpoint.

This update to that manual incorporates changes to the z/OS environment since the original publishing that allow for increased availability, and new or changed commands that allow for more dynamic changes. It also includes the major subsystems that have matured in the last decade that allow for greater availability. The book includes links to other manuals for further information.

This book tells you of some of the changes and enhancements that are available and how to find out more about them. Take into account the releases of software you are running and the environment you are currently working in.

The book focuses on the enhancements from a systems standpoint. It separates and uncouples systems from applications. This book does not cover application design, deployment, or tuning.

The intention is to provide the tools to build a more resilient environment both from an operating systems and subsystem standpoint, and to describe how to separate an IPL of one LPAR from the subsystems running on that LPAR.

The book is broken down into the following chapters:

- ▶ Chapter 1 - The current chapter
- ▶ Chapter 2 - How to start planning for increased IPL reduction windows, what windows are correct for you, and some experiences from clients who have gone through IPL reductions
- ▶ Chapters 3-17 - z/OS from 1.8 - 2.1 with major availability enhancements outlined along with some new and improved commands that allow for dynamic changes
- ▶ Chapter 18 - CICS and how to keep it functioning and available while accomplishing housekeeping, reorgs, and so on
- ▶ Chapter 19 - DB2 and how to keep it functioning and available while doing online reorgs and other changes
- ▶ Chapter 20 - Describes improvements in IMS resilience and explains what requires a recycle
- ▶ Chapter 21 - WebSphere MQ talking to the client and how to keep the conversation going without a recycle
- ▶ Appendix A - What still requires recycling and why? Some changes to the original list of things that could not be done dynamically and what release they were changed
- ▶ Appendix B - DB2 UNIX file system maintenance considerations

## 1.8 Live Guest Relocation

Live Guest Relocation (LGR) provides the capability for a running Linux virtual machine to be migrated from one VM system to another without disruption within a VM SSI cluster. Currently, this is only available with IBM z/VM® 6.2 with Single System Image (SSI), multiple (up to 4) VM LPARs sharing common code.

Live guest relocation provides continuity for virtual server workloads over planned z/VM and machine outages, such as hardware upgrades or service. There is a verification process provided that helps ensure that you have the proper resources on your receiving system before moving the workload. This verification process is also available on request so you can start planning for the receiver's eligibility.

For more information, see *z/VM V6.3 General information*, GC24-6193, and *An Introduction to z/VM Single System Image (SSI) and Live Guest Relocation (LGR)*, SG24-8006.



## Project planning and user experiences

This chapter describes how you can undertake a project to reduce the number or frequency of IPLs. It provides the combined experiences and insights of two example enterprises and describes some of the challenges they encountered and how those challenges were addressed. In addition, it addresses the question of identifying the “right” IPL frequency for your company.

**Note:** This chapter assumes some knowledge of the components and middleware in z/OS. Because those are covered in the following chapters, you might want to come back to this chapter after reading the rest of the book.

This chapter includes the following sections:

- ▶ Project planning considerations
- ▶ Setting the rules
- ▶ Teams to involve
- ▶ Researching and developing a strategy
- ▶ Changes and how to control them
- ▶ How frequently should I IPL or restart my subsystems?
- ▶ Systems Management considerations
- ▶ User experiences

## 2.1 Project planning considerations

Enterprises are finding that they are increasingly dealing with clients and users all over the world, in different time zones, with different expectations of availability, different maintenance windows, and different service level agreements (SLAs).

A number of years ago, a large financial institution initiated a project to examine and optimize their IPL frequency. This project was in response to feedback from the business community that the then-current practice of weekly IPLs was having a negative and unacceptable impact on their business. The ultimate objective of this project was to provide higher availability to the business globally.

A small committee of key people was formed with following objectives:

- ▶ Clearly identify the short and long-term goals of the project.
- ▶ Identify which teams needed to be involved.
- ▶ Decide how much funding was required to cover the costs of investigating the situation and to implement possible resolutions.
- ▶ Investigate the current IPL practices and the reasons for each IPL based on data such as the IPL schedules, outage moratoriums, SLAs, and so on.

After the goals were determined and agreed upon, an impartial Project Manager was appointed. Additionally, it was agreed early on in the project that involving people from all the different areas involved would result in fewer things being missed and fewer problems (both technical and otherwise) going forward.

## 2.2 Setting the rules

First, some basic rules had to be agreed on that determine how to proceed. The initial committee of people met to lay out the project parameters:

- ▶ A Project Leader to keep score, act as a referee, document commitments, and follow up on questions
- ▶ Involve teams and get a representative assigned from each team to attend all meetings (see 2.3, “Teams to involve” on page 11 for a suggested list of teams to involve)
- ▶ Recognize that everyone’s vote is equally important
- ▶ Take things slowly

There is an old saying that “If you want to go fast, go alone. If you want to go far, go with others.” The client felt that this perfectly described their experience with this project.

## 2.3 Teams to involve

To ensure a smooth and fault-free transition to the new planned outage frequency, it is critical that you have the support and buy-in of all affected groups. The precise teams vary from company to company. However, the following list might be helpful when you are considering who to involve in your project. With the list are some reasons why each group would be involved:

- ▶ Management (If you do not have the support of the affected management, there is little point in proceeding)
- ▶ z/OS
  - Procedures had to be developed by the z/OS team to deal with dynamic changes to assure everyone that the changes would not be regressed with the new IPL schedule.
  - All the changes that could NOT be done dynamically had to be identified and published to the mainframe support group so that proper planning could be done and changes implemented in a timely fashion.
  - An IPL schedule had to be developed to be published and change control records had to be modified to incorporate the new IPL schedule.
  - Weekly IPLs were used to put on maintenance and install new releases. These had to be planned further in advance and coordinated with the new IPL schedule.
- ▶ Hardware
  - All changes that could not be done dynamically had to be identified and published.
  - All dynamic changes had to be tested and documented. After the dynamic changes were implemented, they were checked for accuracy.
  - New hardware (CPUs, storage, tape systems) had to be planned out and coordinated. In some cases exceptions to the IPL schedule had to be requested and approved by everyone.
- ▶ Infrastructure software (for example, ISV products)
  - New releases of programming products had to be planned out carefully and requirements listed (new linklist, lpa, svc, ssn, and so on). If necessary, new releases were coordinated with the IPL weekends should the requirements not be conducive to dynamic change.
  - Upgrades of programming products had to be examined for timing and requirements.
- ▶ Transaction manager systems (CICS, IMS TM, WebSphere Application Server, and so on)
  - Weekly IPLs were used to put on maintenance and install new releases. These had to be planned further in advance and coordinated with the new IPL schedule.
- ▶ Database manager systems (DB2, IMS, ISV database managers, and so on)
  - Weekly IPLs were used to put on maintenance and install new releases. These had to be planned further in advance and coordinated with the new IPL schedule.
- ▶ WebSphere MQ systems
  - Weekly IPLs were used to put on maintenance and install new releases. These had to be planned further in advance and coordinated with the new IPL schedule.
- ▶ File Transport (such as NDM, FTP, and so on)
  - Because this particular client depended on File Transport to deal with their many customers, this team was particularly important to keep in the loop of any significant changes, such as IPL schedules that might affect their products. Maintenance

windows, new installs, and housekeeping all had to be researched and procedures established to deal with anything that could not be handled dynamically.

- ▶ Asset Management
  - Because many groups depended upon weekly IPLs to schedule license changes or upgrades that required license changes, Asset Management had to be kept informed of the new IPL schedule so they could handle any requests for new product licenses.
- ▶ Automation
  - The Automation team was tasked with coming up new routines to recycle various subsystems that normally were recycled with an IPL. This all had to be thoroughly tested and monitored.
- ▶ Batch scheduling product (IBM Tivoli® Work Scheduler, for example)
  - The Batch scheduler was normally recycled weekly with the IPL to close log files and do any clean-up work. A new procedure had to be developed to recycle the scheduler weekly and uncouple it from its dependency on an IPL.
- ▶ Operations
  - Operations had to get accustomed to NOT running an IPL on everything every week and to adhere to a new IPL schedule. There was a published IPL schedule and a change control record that was sent to Operations mid-week so they would be aware of the systems that were going to go through IPL. If there were any changes to that schedule, they had to be aware of them going forward.
  - Operations also was on the front line of any alerts that might result from a reduced IPL schedule (logs filling up, alerts that are generated when something goes wrong) so that they can quickly notify the appropriate team.
- ▶ Applications
  - Applications, who work closely with the subsystems groups and the z/OS team, had to be notified of the IPL schedule change because they often were the original source of change requests to be implemented with an IPL.
- ▶ Network
  - The Network team, as with others, depended on weekly IPLs to pull in changes to IBM VTAM®, Session-manager type products, and general network changes. These had to be identified and scheduled on a more timely basis. In some cases, they had to uncouple the changes from the IPL schedule, and in others they had to plan for IPL weekends to install
- ▶ Capacity Planning and Performance
  - Capacity Planning often was the source of requests against a particular logical partition (LPAR) or group of LPARS that required IPLs. These requests had to be coordinated with regards to the IPL schedules. LPAR moves became part of a larger plan that was scheduled well in advance and that tried to adhere to the IPL schedule in place.
- ▶ Potentially other teams, depending on your software portfolio and your corporate and IT structure.

**Note:** The order of teams in this list is not intended to reflect the relative importance of the various groups. Some groups are more affected by the change than others, but the support of all groups is equally important.

**Note:** Because the IPLs were only being done monthly in this particular case, many changes were scheduled for the same IPL by many different groups. Care had to be taken to coordinate all the changes among the various groups to ensure no one got in anyone else's way while accomplishing their own goals. To this end, a weekly mainframe meeting, which in the past had only been used to describe z/OS changes, was expanded to include all changes going on for the weekend and the near future. This process helped ensure that everyone was aware of all changes and was able to speak up if they saw a problem. Some shops appointed a "change manager" or "availability manager" to coordinate and communicate activity.

## 2.4 Researching and developing a strategy

In the first meeting, the z/OS team was tasked with going back a year and documenting all the IPLs, identifying the reasons for each, and grouping the outages by reason.

In subsequent meetings, the resulting spreadsheet was reviewed by the entire team. One of the first actions of the team was to determine whether certain IPLs were being carried out because "we always did it that way", and therefore potentially could be avoided. Obviously, if possible, the event or situation that caused the practice to be started in the first place should be identified and investigated to determine whether it still existed.

Discussions were then held with application owners to determine what levels of availability they needed and why, and what level they were willing to live with or tolerate.

Discussions were also held with the infrastructure teams (z/OS, hardware, storage, network, and so on) to determine what frequency of planned outages they felt would be acceptable.

After determining the application and infrastructure requirements, a proposal was presented to management. The proposal included performing further research, followed by testing, and possible implementation. The required approval and funding was secured, and the project commenced.

The team then tackled the next step: How would they test the impact of the reduced IPL frequencies? They were fortunate to have a test sysplex that contained a couple of systems that could be dedicated to this project. They left these two LPARs up for stints of 4 to 6 weeks without running an IPL. During this time, they performed all the PARMLIB and other changes that could be done dynamically, and scheduled any remaining changes for the next IPL.

After they were able to prove that they could survive extended periods of time without an IPL on the test LPARs, they proceeded to the next step. The client had two members of the sysplex that were used solely as communications LPARs. Because the communications LPARs do not run CICS, DB2, or any of the other major subsystems that require care and feeding, they were good candidates for a proof of concept where everyone could get comfortable with the increased time between IPLs. During the time these LPARs were up without an IPL, the team met weekly to describe any problems that were encountered and to determine the next step.

The next step was to gradually introduce production LPARs to the limited IPL schedule and monitor them closely. This turned out to be more difficult than the other steps. It transpired that many subsystem owners were relying on the weekly IPL to drive a weekly restart and counted on that to do clean up work, close and process log files, and perform general housekeeping. Working with the Scheduling and Automation groups, they gradually implemented weekly restarts for the subsystems where the owners still required it. This

actually took quite a bit of time and negotiation, and LPARs were carefully chosen and modified to accommodate the reduced IPL schedule.

The increased time between IPLs created a number of unanticipated issues. However, there were no show-stoppers, and each issue was addressed as it was encountered. The most important rule was that the changes should not have any negative impact on the availability of the production applications.

In the test environment, they were able to go directly to monthly IPLs for testing purposes. In the production world, the communications LPAR IPLs were first moved to bi-weekly IPLs. When that change proved to be acceptable and issues had been addressed, the IPL frequency was adjusted to be monthly. In today's System z environment, running an IPL quarterly (or even less frequently) is a viable goal.

## 2.5 Changes and how to control them

One of the corollaries of the reduced frequency of IPLs was that many changes would have to be done dynamically. This was a major concern to many people. Would dynamic changes be lost between IPLs? How would they be documented? Who would control them?

An important lesson that the client learned is that you cannot have too many backups. The client has a methodology for keeping a backup (IPLable) copy of all changes to the SYS1.xx.PARMLIB data set. They take nightly backups of critical data sets including SYS1.PARMLIB and SYS1.xx.PARMLIB that can always be referred to. The backups also provide a valuable audit trail if something goes amiss.

Rules were set in place that dictated that all changes must be “hardened in” before being dynamically applied to the system. They limited access to who could actually make the changes. Update access to SYS1.xx.PARMLIB was limited to the person responsible for changes that week and that was the same person who was issuing the dynamic activation commands and coordinating the testing with the requester.

Documentation became even more important, and the standards dictated that any change had to have an entry in a documentation member in the library. The entry included this information:

- ▶ The reason for the change
- ▶ What the change was expected to achieve
- ▶ The person responsible for the change
- ▶ The member that was changed
- ▶ The date that the change would be implemented
- ▶ One item that was critical and documented in each change control record was who would be doing the testing for each change, not necessarily the implementor but the requester working with the implementor.



To always have a good usable backup of a particular member, a backup member could only be created once between IPLs or dynamic activation. When an active member had been used in an IPL or dynamically activated, it could then be used as the basis for another change. The following is an example of this process:

- ▶ Assume that you need to make a change to the PROG01 member in SYS1.xx.PARMLIB for this weekend.
- ▶ You create your backup, in this case PROG02, then make your change to PROG01, and document the change in the \$\$DOC member.
- ▶ The updated PROG01 member is then activated by using the **SET PROG=01** command.
- ▶ You are now finished for this weekend. If the change had been unsuccessful, it could be backed out by reverting to the PROG02 member.
- ▶ In the next week, you must make another change to the PROG01 member. So, once again, you create a backup by copying PROG01 into PROG02. It is acceptable to overwrite the old PROG02 member because the to-be-updated PROG01 member has already been activated and tested in production.
- ▶ Make the change to the PROG01 member and document it.
- ▶ Dynamically activate the change by using the **SET PROG=01** command.
- ▶ The following weekend is an IPL weekend and the current PROG01 member with all the dynamic changes you have created will be used at IPL time. Because changes are saved to the member that will be used for the next IPL before the change is implemented, you are assured that dynamic changes are not lost.

Some clients choose to run all their **SET** commands in batch so they can keep an audit trail of their changes to compare week to week. As long as you have a methodology for controlling your changes that works for *you*, you will be fine.

For changes to a member that does not support dynamic refresh (the ALLOCxx member, for example), you need to allow for the changes being regressed by the next IPL if they are not hardened to the corresponding member. If the change is to be considered a permanent change, the appropriate member (in this case ALLOCxx) must be updated and a note must be made in the documentation that a **SETALLOC** was issued. Even if the change was only for a specific moment in time (from one IPL to the next), make a note in the documentation that the command was issued, what it did, and how long it was expected to last. If the **SETxxx** command is issued as a test or to cure a temporary solution, that should be noted as well so that you always know what your current environment looks like as opposed to what is documented in PARMLIB.

Additionally, because the member is not being refreshed, the syntax will NOT be checked until that member is used for the next IPL.

A few PARMLIB members support both types of refresh. For example, to change some SMF parameters, you can update the SMFPRMxx member and issue a **SET SMF=xx** command. Alternatively, you can use the **SETSMF keyword=value** command to change just a single setting.

For this type of member, you need to decide on how changes will be implemented and ensure that everyone conforms to those rules. Although these members provide great flexibility, you need to be extra careful that changes made by using the **SETxxx keyword=value** process do not get regressed at the next IPL.

For more information about which PARMLIB settings can be changed and by which method, see *z/OS MVS System Commands*, SA38-0666.

## Change planning

Because of the increased duration between planned outages, changes also had to be planned much further in advance. Upgrades to subsystems, program products, operating systems, and maintenance of all these products had to be thought through and scheduled well in advance. A published IPL schedule became everyone's best friend and many copies were printed off and posted in cubicles and public spaces. Hardware changes that still required an IPL or Power-On-Reset had to be described thoroughly and sometimes had to become exceptions to the limited IPL schedule.

Performance and Capacity Planning, in addition to being involved in LPAR moves and CPU changes, also had to plan well in advance for various performance issues that might creep up during a longer period between IPLs. Many storage issues are resolved with an IPL and counting on a weekly IPL (or not thinking about it at all) needed to be addressed. Planning ahead and suggesting tuning parameters became part of the project. A good tool that can be used for forecasting is Predictive Failure Analysis (PFA), which can be run outside of Health Checker. More information can be found in 3.5.15, "Predictive Failure Analysis (PFA)" on page 55.

One important thing that was learned was that everyone had to be flexible and accommodating. If one group encountered a problem as a result of the less frequent IPLs, it was possible that the same issue could affect another group. This fostered even greater cooperation between the groups because everyone benefited from the shared knowledge.

## 2.6 How frequently should I IPL or restart my subsystems?

A common question from clients is "what is the recommended IPL frequency?" It depends because the ideal IPL frequency can be different for each company, depending on their availability requirements, their workloads, their software mix, the volatility of their environment, their uptake of new technology, and many other variables.

It is generally wise to avoid extremes. Running an IPL on a system every week should not be necessary. Similarly, running for a long time is also not a good idea, because you will not be able to apply preventive service.

But before you get too deeply into trying to identify the best IPL frequency, step back and consider some things:

- ▶ If application availability is critical to your business, use IBM Parallel Sysplex data sharing and dynamic workload routing. These allow you to disconnect application availability from system availability, which gives you vastly more flexibility regarding how frequently you can IPL a system. Remember that Parallel Sysplex data sharing is not only intended to protect you from unplanned outages, but it also masks planned outages from your application users.
- ▶ Many aspects of z/OS, its subsystems, and the underlying hardware support dynamic change. The list of items that still require a planned outage is getting smaller all the time. Get away from the mindset that change equals outage. There are still reasons why you need to restart a system or subsystem, but try to plan your IPL frequency strategy around things that *require* an outage.

If you can achieve these things, then you can more easily identify what is truly the optimal IPL frequency for you. If you are *not* able to achieve these things, the IPL frequency is likely to be determined by these non-technical issues:

- ▶ Deciding what application interruptions are acceptable to your users and business requirements
- ▶ Performing IPLs or subsystem restarts that are not really necessary (the “we have always done it this way” syndrome)
- ▶ Not using the features that are available to avoid planned outages

Having considered these things, if you still determine that you want to change the IPL frequency from your current strategy, start by researching the workloads on each system and the dependencies between the systems. It is wise to start with incremental changes.

If you can, begin with some test systems and monitor everything closely. After all interested parties are comfortable with the process for making dynamic changes, move forward to some low impact production systems. If you are more comfortable going to a two-week cycle rather than monthly, do that. Do not do any of this overnight. You did not build your complex environment overnight, and making dramatic changes might result in highly visible problems or outages. This will frustrate all involved and place the project at risk.

A key to the success of the project is knowing your own systems. The client found that everyone on the team learned more about their own applications and how they interfaced with each other than anyone would have imagined, and everyone was better off for it. Teams were able to plan more carefully for upgrades and changes, and had a greater awareness of the possible impact of changes on other teams.

Having changed their IPL frequency from weekly to monthly, this client is now preparing to move to the next stage. This involves moving some systems to quarterly IPLs, and possibly even on-demand IPLs as part of a project to provide near-continuous availability for critical applications.

Since the predecessor to this book (*z/OS Planned Outage Avoidance Checklist*, SG24-7328) was published in 2006 using z/OS 1.7, great strides have been made in the area of high availability. The ability to dynamically activate changes to PARMLIB settings has been enhanced. z/OS UNIX has added support for dynamic updates and restarts. DB2 has been greatly enhanced to support dynamic changes to the DB2 subsystem and the data schema. All of these enhancements must be planned for carefully and tested thoroughly. The advent of many of these options can mean that the user need never know that a system has undergone IPL or that a subsystem was recycled.

## 2.7 Systems Management considerations

As was mentioned in *z/OS Planned Outage Avoidance Checklist*, SG24-7328 (and still true today), the way you manage your systems can have a fundamental impact on how you control and manage planned outages.

Planning ahead is critical:

- ▶ You might need to install new hardware or software
- ▶ You might need to apply service to the operating system or its components
- ▶ You might need to address storage fragmentation or “storage creep”

Careful planning can help you postpone or eliminate a planned outage. Hardware plan-ahead activities can help you avoid power-on resets (PORs). Prudent setting of system parameters

can forestall IPLs by (for example) adjusting the size of the ECSA to accommodate storage leaks until they can be resolved. Strong systems management also enables you to monitor the use of system resources and extrapolate the point at which they will be exhausted.

## 2.8 User experiences

When planning for an increased time between IPLs, the things that immediately come to mind are storage fragmentation, running out of non-reusable address space IDs, and so on. These are all valid concerns. However, you must also consider other things. This section includes some of the considerations were encountered by these clients.

### 2.8.1 Batch schedulers

When the client moved to monthly IPLs, they found that because of the volume of batch jobs, the logs that store all the activity of the batch scheduler were enormous. As a result, searching for issues in the message logs took far too long. To address this issue, they instituted a weekly recycle of the scheduling controller.

### 2.8.2 Session manager products

The client found after a month that the session manager product was using a large percentage of spool space. The session manager procedure was changed to route its output to an archiving product.

In this case, the offender was the session manager. However, the same concern applies to any job or started task that is normally started after the IPL and not stopped until the next IPL *and* that produces large amounts of spool output.

To identify such tasks, compare the list of tasks that are started at IPL to the list of tasks that are stopped as part of the system shutdown procedure. Then, check to see how much spool space is used by each of those tasks. For those that produce a large amount of output, consider using the SPIN capabilities (described in 13.3, “Early release of job log from long running jobs and started tasks” on page 152 and 13.4, “Use SPIN=UNALLOC JCL keyword to spin off spool files” on page 153).

### 2.8.3 Tasks using Getmain and Freemain

When they increased the duration between IPLs, the client found that a number of tasks ran out of region because they were issuing a GETMAIN without a corresponding FREEMAIN. Because the tasks were previously being restarted every week, the storage use was not growing fast enough to run out storage within the week. However, when the task was potentially running for four times longer, they did exhaust their storage before the next IPL.

The situation is a bug and should be addressed with the vendor of each application. For information about how to debug a situation like this, see the description of the DIAGxx member in *z/OS MVS Initialization and Tuning Reference*, SA23-1380. Also, see the information about GFS trace in *z/OS MVS Diagnosis Tools and Service Aids*, GA32-0905.

## 2.8.4 Log files and data sets

When certain tasks are stopped during the shutdown/IPL process, logs spin off to GDGs, allowing users to view these logs by pointing to the current GDG number (0). For tasks that now stay up for many weeks, it is a long time before users can view the output. Worse still, if users are used to getting last week's data from the (0) generation, that data set returns data that are up to four weeks old.

This particular issue could affect any long running job or task that does logging. Each one must be examined individually for resolution. One option is to change the program to close and reopen the file at a specific interval. Another option might be to send the output to a spool data set and use the SPIN function to make the output available sooner.





# Part 1

## z/OS components

This part of the book covers the components of the z/OS operating system that contribute to your ability to plan outages. Each component has a separate chapter:

- ▶ z/OS enhancements: Availability and diagnostics
- ▶ z/OS enhancements: Base Control Program
- ▶ z/OS enhancements: GRS
- ▶ z/OS enhancements: Resource Recovery Services
- ▶ z/OS enhancements: XCF and System Logger
- ▶ z/OS enhancements: Communications Server
- ▶ z/OS enhancements: DFSMS
- ▶ z/OS enhancements: Distributed File Service (zFS)
- ▶ z/OS enhancements: Security Server (RACF)
- ▶ z/OS enhancements: ICSF
- ▶ z/OS enhancements: JES2
- ▶ z/OS enhancements: JES3
- ▶ z/OS enhancements: Infoprint Server
- ▶ z/OS enhancements: TSO/E
- ▶ z/OS enhancements: z/OS UNIX







## **z/OS enhancements: Availability and diagnostics**

This chapter and subsequent chapters provide information about enhancements in z/OS from Version 1 Release 8 (1.8) that contribute to its availability.

This chapter includes the following sections:

- ▶ List of z/OS enhancements
- ▶ PARMLIB management for availability
- ▶ Dynamic exit support for availability
- ▶ Scalability for availability
- ▶ Diagnostics for availability

## 3.1 List of z/OS enhancements

Table 3-1 lists the enhancements that are covered in this book. It is useful as a checklist to ensure that an installation is using all of the appropriate enhancements.

Table 3-1 Checklist of outage avoidance items for z/OS

Feature/Enhancement	Used?
3.2, "PARMLIB management for availability" on page 27	
3.3, "Dynamic exit support for availability" on page 30	
3.4, "Scalability for availability" on page 32	
3.5, "Diagnostics for availability" on page 35	
4.2, "Change nearly any value in ALLOCxx without an IPL" on page 59	
4.3, "More effective use of space in ECSA" on page 60	
4.4, "PROG enhancement: Dynamic LNKLIST" on page 62	
4.5, "PROG enhancement: Dynamic LPA" on page 64	
4.6, "PROG enhancement: Dynamic exit replacement" on page 65	
4.7, "SETPROG enhancement: SVC enhancement" on page 66	
4.8, "Enhancements to LLA processing" on page 67	
4.9, "System REXX availability enhancements" on page 68	
4.10, "Dynamically change system symbols" on page 70	
4.11, "Add more than RESERVED number of CPs without an IPL" on page 73	
4.12, "Switching between primary and secondary DASD without an IPL" on page 75	
4.13, "Specify NOBUFFS action (SMF) at the log stream level" on page 76	
4.14, "Forcing hung commands to avoid an IPL" on page 77	
4.15, "Dynamically configure storage-class memory (SCM)" on page 78	
4.16, "End a task with new operand on FORCE" on page 79	
4.17, "Auto-reply for WTOR" on page 80	
4.18, "DCCF support for WTOR Auto-Reply" on page 81	
4.19, "New MODIFY VLF command" on page 83	
4.20, "Add/Remove MCS consoles dynamically" on page 84	
4.21, "New Message Flood Automation" on page 85	
4.22, "Added LPA defer wait capability" on page 87	
4.23, "Auto-IPL using DIAGxx" on page 88	
4.24, "z/OS reusable address space support and usage" on page 89	
4.25, "z/OS report on linkage indexes" on page 93	
5.1, "Moving GRS contention notification system role" on page 96	

Feature/Enhancement	Used?
5.2, "Control the maximum number of ENQs in a system" on page 97	
5.3, "Changing to GRS without sysplex IPL" on page 98	
6.1, "RRS internal restart" on page 100	
6.2, "Clean RRS shutdown option" on page 101	
6.3, "Ability to dynamically turn on and off RRS archive log" on page 102	
7.1, "System Status Detection Partitioning Protocol" on page 106	
7.2, "CRITICALPAGING function for HyperSwap environments" on page 107	
7.3, "Non-disruptive CF dump" on page 108	
7.4, "XCF command to get structures to primary CF" on page 109	
7.5, "Force disconnection or deletion of a log stream without restarting the Logger address space" on page 110	
7.6, "IXGCNFxx PARM" on page 111	
7.7, "System Logger threshold messages" on page 112	
7.8, "Multiple logstream/data set management" on page 113	
7.9, "Dynamic log stream updates for duplexing" on page 114	
8.1, "Specify a hot standby server to Sysplex Distributor" on page 116	
8.2, "Resolver start error tolerance" on page 117	
8.3, "Command to verify TCP profile syntax" on page 119	
8.4, "RPCBIND/NFS re-registration" on page 120	
9.1, "Dynamically activate changes to DEVSUPxx member" on page 122	
9.2, "Dynamically modify size of SMSPDSE hiperspaces" on page 123	
9.3, "VSAM CA reclaim for KSDSs" on page 124	
9.4, "Support for very large catalogs and catalogs on extended address volumes (EAVs)" on page 125	
9.5, "Minimizing application disruption during catalog maintenance" on page 126	
9.6, "DADSM and CVAf support for dynamic corrective service activation" on page 127	
9.7, "DFSMSHsm Control Data Set backup enhancements" on page 128	
9.8, "Allow XRC primary volumes to be offline when the XSTART and XADDPAIR commands are issued" on page 129	
9.9, "Metro Mirror (PPRC) support for Remote Pair FlashCopy" on page 130	
9.10, "Remove need to offline/online volume after VTOC change" on page 132	
9.11, "Dynamically update maximum number of tape drives OAM allocates" on page 133	
10.1, "zFS internal restart" on page 136	
10.2, "zFS automatic re-enable of disabled file systems" on page 137	
11.1, "Sample passphrase ICHPWX11 exit uses System REXX" on page 142	

Feature/Enhancement	Used?
11.2, "Sample password ICHPWX01 exit uses System REXX" on page 142	
12.1, "ICSF: Improve product stability" on page 146	
13.1, "Dynamic JES2 exit support" on page 150	
13.2, "Dynamic changes to JES2 parameters" on page 151	
13.3, "Early release of job log from long running jobs and started tasks" on page 152	
13.4, "Use SPIN=UNALLOC JCL keyword to spin off spool files" on page 153	
13.5, "Ability to stop a JES2 job at the end of the current step" on page 154	
13.6, "Dynamically modify a JES2 spool volume" on page 154	
13.7, "Support for over four billion spin data sets" on page 155	
14.1, "Dynamically modify TCP/IP NJE information" on page 158	
14.2, "Ability to dynamically add spool volumes to JES3" on page 159	
14.3, "Ability to dynamically remove spool volumes from JES3" on page 161	
15.1, "Dynamic configuration changes for most InfoPrint options" on page 166	
15.2, "Print Services Facility (PSF) installation is disassociated from SYS1.NUCLEUS" on page 167	
16.1, "Support for VTAM unconditional reconnect for TSO/E users" on page 170	
17.1, "Support dynamic modification of AUTOCVT setting" on page 174	
17.2, "Prevent content overlay during mount" on page 174	
17.3, "Change sysplex root data set without sysplex IPL" on page 177	
17.4, "Support for alternate sysplex root data set" on page 178	
17.5, "Change number of common inet ports without OMVS restart" on page 179	
17.6, "Remount with same mount mode" on page 180	
18.2, "Techniques to reduce unavailability through configuration changes" on page 184	
18.3, "Reduction of preventive recycles of CICS" on page 197	
18.4, "Reducing shutdowns through runtime dependencies" on page 207	
18.5, "Reducing shutdowns to allow resource access by non-CICS tasks" on page 209	
18.6, "CICS still must be stopped, so what next?" on page 214	
19.1, "DB2 subsystem or data sharing group member outage" on page 216	
19.2, "Database changes" on page 228	
20.1, "IMS database considerations" on page 252	
20.2, "IMS Transaction Manager Considerations" on page 256	
20.3, "IMS system considerations" on page 257	
20.4, "Resources still requiring a recycle of IMS to change" on page 261	
21.2, "Availability and planned outages" on page 265	
21.3, "Applying maintenance during subsystem operation" on page 269	

Feature/Enhancement	Used?
21.4, “Changes that can be made dynamically” on page 272	
21.5, “Changes requiring a queue manager restart or IPL” on page 273	
21.6, “Error recovery (without a system or subsystem restart)” on page 274	

## 3.2 PARMLIB management for availability

This and subsequent chapters describe making dynamic changes to your systems. But dynamic change can have both desirable and undesirable consequences:

- ▶ Changing something dynamically can avoid a planned outage (desirable).
- ▶ Changing something dynamically introduces the risk that the change will be regressed when you next restart your system or subsystem (undesirable).

Particularly in this chapter, many of the enhancements described are related to system settings that are controlled by one of the members in the system PARMLIB concatenation. It is always a good practice to have a set of conventions for managing your PARMLIB members that is known and adhered to by all. However, in an environment where dynamic changes are being made, comprehensive PARMLIB member management conventions combined with strict change management are a necessity.

There is no one “right” way to manage your PARMLIB members. The correct convention for *your* installation is one that meets all your requirements, is easily understood, and is adhered to by all. Regardless of what convention you use, it must meet the following requirements:

- ▶ It must always be possible to back out to a known working prior point.
- ▶ The member names should be such that there is never any doubt about which member is used to roll *in* a change, and which member is used to *back out* a change.
- ▶ It must allow for the possibility that different changes might be in process at the same time.
- ▶ It should provide for the situation where you want to implement some or all of those changes. Similarly, it should provide the ability to back out some, but not all, changes.
- ▶ It should allow for the fact that different groups might need access to the PARMLIB members. For example, the CICS support group need to have entries in some PARMLIB members. You might want to have separate sets of PARMLIB members for each support group.
- ▶ It should provide for the fact that many dynamic changes might be made in the time period between successive IPLs.
- ▶ It should work with members that support refresh using the **SET** command (for example, **SET SMF=xx**). In this case, you update the member and then activate the change by issuing the **SET SMF=xx** command.
- ▶ It should work with members that support dynamic updating of individual keywords (for example, **SETSMF FLOOD(ON)**). In this case, your process must provide for the fact that activating the change is a separate process from updating the associated PARMLIB member.

Most PARMLIB members support just one update mechanism; either the update-a-single-keyword model, or the refresh-the-whole-member model. But a subset support both models. For those members that support both models, you need to decide

whether you to make the dynamic changes by just changing the one keyword, or by updating and refreshing the member.

An advantage of the second option (assuming that you updated the member that will be used for the next IPL) is that you do not have to worry about regressing the change at the next IPL. Another advantage is that by activating your change by refreshing the member ensures that the syntax is correct, rather than doing an IPL and finding then (when the system is only half up) that you have a problem. The disadvantage is that you might inadvertently also pick up other changes that have been applied to the member you refresh from.

Which option you select is related to your confidence in your change management process. The first option might be suitable for some clients, and others might prefer the second. In any case, make this change management decision deliberately and ensure that it is adhered to by all your support personnel.

Unfortunately, no single PARMLIB member syntax checker works with all members. However, a number of tools provide assistance in this regard. The following sections describe many of them.

### 3.2.1 Syntax checker: Language Environment

For the IBM Language Environment® PARMLIB members, you can use an Language Environment-specific syntax checker using the following JCL:

```
//CEEPRMCJ      EXEC PGM=CEEPRMCC,  
//              PARM='CEE=(xx,yy,...,nn)'  
//CEEPRMCK      DD DSN=MEENAK.SYSTEM.PARMLIB,DISP=SHR
```

### 3.2.2 Syntax checker: z/OS UNIX

Unlike the Language Environment members, there is no batch program to check the syntax of the z/OS UNIX members. However, the **SETOMVS SYNTAXCHECK** command can be used to validate the format of the BPXPRMxx member.

Also, the USS\_PARMLIB check provided by the IBM Health Checker for z/OS determines whether there are differences between the current system settings and the settings defined in the BPXPRMxx member. If a difference is found, an exception message is issued. You receive a report that lists the differences.

### 3.2.3 Syntax checker: TSO/E

The TSO **PARMLIB** command lets you list and update TSO/E specifications that are in effect on the system. Those TSO/E specifications include tables of authorized commands and programs, and default values for some TSO/E commands.

The CHECK function of the **PARMLIB** command lets you check the syntax of any IKJTSOxx member of SYS1.PARMLIB, including active members.

### 3.2.4 Syntax checker: BCP PARMLIB Processor

The PARMLIB Processor allows for the verification of symbolic substitutions without doing an IPL. The Parser is in SYS1.SAMPLIB and is activated as follows:

```
TSO EX 'SYS1.SAMPLIB(SPPINST)' '''SYS1.SAMPLIB(SPPPACK)'''
```

This command creates and populates four data sets, where `myid` is the value specified on the TSO PROF PRE(myid) command:

```
myid.PARMLIB.EXEC  
myid.PARMLIB.PANELS  
myid.PARMLIB.MESSAGES  
myid.PARMLIB.NOTES
```

After the data sets are created, issue the following command:

```
TSO EX 'myid.PARMLIB.EXEC(SYSPARM)'
```

The `myid.PARMLIB.NOTES` file, member `WORKFLOW`, gives an overview of the tool's capabilities:

- ▶ LOADxx processing
- ▶ PARMLIB symbolic preprocessing
- ▶ PARMLIB member selection list
- ▶ Obtaining a PARMLIB member selection list from a concatenation of PARMLIBs

**Note:** The old GRSRNL syntax checker was replaced when Wild Card '\*' logic was added.

The tool has several limitations:

- ▶ The tool does not support editing of `SYSn.IPLPARM` data sets. The tools can be used to generate the member in `SYS1.PARMLIB` or `high_level_qualifier.PARMLIB`. A copy of the member can be placed into the appropriate `SYSn.IPLPARM` data set.
- ▶ The tool assumes that any `SYSn.IPLPARM` data set is on the same volume as the specified `PARMLIB` data set. It does not support the concept of a separate IODF and `SYSRES` volume, nor does it use Master Catalog to locate a `SYSn.IPLPARM` data set.
- ▶ Although the code can detect the presence of a VSAM data set, it cannot read one. Therefore, the IODF data set is only checked for existence. Its contents are not verified.
- ▶ The Master Catalog information is not verified.
- ▶ When in EDIT mode, the data in a member is not altered until it is saved. If running in split screen mode, the effects of the change are not seen until the EDIT mode changes are saved.

For more information, see the Symbolic PARMLIB Parser Appendix in *z/OS MVS Initialization and Tuning Reference*, SA23-1380

### 3.2.5 Syntax checker: JES3

The JES3 initialization stream checker utility tests your JES3 initialization statements before you run a hot start with refresh, warm start, or cold start of JES3. The initialization stream checker detects most syntax errors and some logical errors in the initialization stream. You can run this utility as a batch job or as a TSO job using a command list (CLIST). You must complete two steps to use the initialization stream checker:

- ▶ Step 1 gathers data for the initialization stream checker.

Step 1 obtains hardware configuration data that the initialization stream checker uses to detect logical errors in the `DEVICE`, `HWSNAME`, `RJPLINE`, and `SETNAME` initialization statements. If you omit this step, the initialization stream checker performs only syntax checking.

- Step 2 runs the initialization stream checker.

Step 2 runs the initialization stream checker. The initialization stream checker examines all initialization statements for correct syntax, except the DYNALLOC statements, and creates the JES3 intermediate initialization tables. The initialization stream checker detects logical errors in the DEVICE, HWSNAME, RJPLINE, and SETNAME statements by comparing the JES3 initialization data with the configuration data that you obtain in step 1. For more information, see *z/OS JES3 Initialization and Tuning Guide*, SA32-1003.

### 3.2.6 Syntax checker: Other products

There are key members in other products that have a syntax check. For more information about the See the Comm Server Chapter - Command to verify TCP profile syntax, see *z/OS IBM Communications Server: IP System Administrator's Commands*, SC27-3661.

Information about the IMS syntax checker, which is an ISPF Application, can be found in *IMS Version 13 Installation*, GC19-3656

## 3.3 Dynamic exit support for availability

The Dynamic Exits Facility uses the EXIT statement of the PROGxx PARMLIB member. Use the EXIT statement type to specify statements that do the following tasks:

- Add exit routines to an exit
- Replace exit routines for an exit
- Modify exit routines for an exit
- Delete exit routines from an exit
- Undefine implicitly defined exits
- Change the attributes of an exit

You can use the PROGxx EXIT statement to define exits to the dynamic exits facility at IPL. You can use multiple ADD statements to add more than one exit routine to a particular exit.

Previously defined exit definitions can be modified with the PROGxx EXIT statement, the SET PROG=xx operator command, or the SETPROG EXIT operator command by using the following methods:

- The EXIT statement of the PROGxx PARMLIB member. The PROGxx EXIT statement interacts with the PROG=xx parameter of IEASYSxx and the SET PROG=xx command. At IPL, operators can use PROG=xx to specify the particular PROGxx PARMLIB member the system is to use. During normal processing, operators can use the SET PROG=xx command to set a current PROGxx PARMLIB member.
- The SETPROG EXIT operator command. This command runs the same functions as the EXIT statement of the PROGxx PARMLIB member. See *z/OS MVS System Commands*, SA38-0666, for more information about the SETPROG EXIT command.
- The CSVDYNEX macro. The CSVDYNEX macro can be used to define exits to the dynamic exits facility, control their use within a program, and associate one or more exit routines with those exits. It can also be used to associate exit routines with the existing SMF and allocation exits, which have been defined to the dynamic exits facility.

For more information about the Dynamic Exits Facility, see *z/OS MVS Installation Exits*, SA23-1381.



### 3.3.1 Dynamic exit support for DADSM IGGPRE00 and IGGPOST0 exits

z/OS V1R13 provides new exit support for the DADSM IGGPRE00 and IGGPOST0 exits.

#### The issue

Before z/OS V1R13, exits for the DADSM pre- and post-processing functions were loaded by DFSMSdfp as installation exits during initialization. They were loaded as modules IGGPRE00 and IGGPOST0. To implement and back out, you had to change the exit, assemble, and then IPL again.

#### How this is addressed in z/OS 1.13

Starting with z/OS V1R13, z/OS dynamic exits services is used to define a pre-processing dynamic exit, IGGPRE00\_EXIT, and a post-processing dynamic exit, IGGPOST0\_EXIT. It associates IGGPRE00 and IGGPOST0 modules as exit routines to these respective dynamic exits. All DADSM functions (create, extend, scratch, partial release, and rename) share these common dynamic exits. They are called where the previous installation exits of IGGPRE00 and IGGPOST0 were called using the same existing interfaces.

This change requires changes to your DFSMSdfp operating procedures and system automation (if any). If you use the IGGPRE00 or the IGGPOST0 installation exits, you do not need to change them in any way; just install them as you always have. DFSMSdfp automatically uses the dynamic exit services and uses your IGGPRE00 or IGGPOST0 installation exit as exit routines to the new IGGPRE00\_EXIT and IGGPOST0\_EXIT dynamic exits. You do not need to change the load module names for IGGPRE00 or IGGPOST0. However, you can change the names if you want to. If you do change the names, update the PROGxx PARMLIB member or issue the SETPROG command to get the modules loaded because DFSMSdfp will not load them as exit routines to the dynamic exits.

You can now have multiple exit routines associated with each of the IGGPRE00\_EXIT and IGGPOST0\_EXIT dynamic exits for the DADSM pre- and post-processing exits. Other programs can use the CSVDYNEX macro to associate their exit routines to these dynamic exits, and can add and delete exit routines from any dynamic exit routine as required. They also can be added and deleted with the PROGxx member of PARMLIB and with the SETPROG ADD operator command. All exit routines are called when the DADSM pre- and post-dynamic exits are called from each DADSM function. The execution of one exit routine might then change the behavior of a subsequent one. The exit routines can be called by the system in any order.

To display exits and status, use this command:

**D PROG,EXIT,EN=CSVDYNEX**

#### Restrictions or considerations

The IGGPRE00 and IGGPOST0 module addresses in the CVAF table (CVFDPR31, CVFDPOR31) continue to be set. Therefore, other programs that continue to use this interface are unaffected. Because dynamic exit services are not used in this case, no other exit routine that is associated with the dynamic exits is called. Change these programs to use dynamic exit services, CSVDYNEX.

#### Rollback

This function is not available in the previous release.

### Further information

For more information about the use of the **PROGxx** member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the **SETPROG** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the **Dynamic Exits Facility**, see *z/OS MVS Installation Exits*, SA23-1381.

## 3.3.2 Dynamic exit support for CSVLLIX1 and CSVLLIX2 (LLA exits)

The z/OS standard Dynamic Exit Facility (implemented in z/OS 1.4) is used by products to reduce the requirement to IPL or to recycle the product task.

LLA exits (CSVLLIX1 and CXVLLIX2) previously were done with Modify commands unique to LLA, such as F LLA,EXIT1(OFF) and F LLA,EXIT1(ON).

- ▶ CSVLLIX1 is LLA Fetch exit
- ▶ CSVLLIX2 is LLA Module Staging exit

### How this is addressed in z/OS 1.12

With z/OS 1.12, the exits can be added to the **PROGxx** exit member and activated with the commands for the dynamic exit facility.

For more information, see “Dynamic Exits Facility” in *z/OS MVS Installation Exits*, SA23-1381”

### Restrictions or considerations

Issue **D PROG,EXIT** to see a list of exits supported.

If **SETPROG** is used to implement dynamic options, you need to update the **PROGxx** member to avoid regressing options at IPL time.

### Rollback

This function is not available in the previous release.

### Further information

For more information about the use of the **PROGxx** member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the **SETPROG** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the **Dynamic Exits Facility**, see *z/OS MVS Installation Exits*, SA23-1381.

## 3.4 Scalability for availability

This section describes two scalability features: EAV and BCPii.

### 3.4.1 Extended Address Volume (EAV)

Before z/OS 1.10, DASD volumes were limited to 65,520 cylinders per volume. To satisfy growing DASD storage requirements, larger capacity DASD were introduced, and many software products had to be modified to fully use these larger volumes.

#### How this is addressed in z/OS 1.10

z/OS 1.10 introduced the extended address volume (EAV), which is by definition any volume that contains 65,521 or more cylinders. The extra space on an extended address volume (the cylinders whose addresses are equal to or greater than 65,536) is referred to as the extended addressing space (EAS). These cylinder addresses are represented by 28-bit cylinder numbers. On an extended address volume, the cylinders whose addresses are below 65,536 are referred to as the base addressing space. These cylinder addresses are represented by 16-bit cylinder numbers or by 28-bit cylinder numbers whose high order 12 bits are zero.

With the old DASD logic, the maximum cylinder used is 65520, but the Track-manage addressability is up to 65536. So an EAV volume uses the Track-manage up to 65536. Therefore, the 16 extra cylinders are processed with the 16-bit structure and the EAS part of the volume is processed with a 28-bit structure.

Extended address volumes are divided into track-managed space and cylinder-managed space. Track-managed space is the space on a volume that is managed in tracks and cylinders. Track-managed space ends at cylinder address 65,519. Each data set occupies an integral multiple of tracks. Track-managed space also exists on all non-EAV volumes.

Cylinder-managed space is the space on the volume that is managed only in multi-cylinder units. Cylinder-managed space begins at cylinder address 65,520. Each data set occupies an integral multiple of multicylinder units (MCU). Space requests targeted for the cylinder-managed space will be rounded up to the next MCU (21 cylinders). The cylinder-managed space exists only on EAV volumes. An EAS-eligible data set is one that can be allocated on cylinder numbers 65,536 or greater.

The following types of data sets were EAS-eligible in z/OS 1.10:

- ▶ SMS-managed and non-SMS VSAM (all types)
- ▶ VSAM data sets inherited from prior physical migrations or copies
- ▶ VSAM temporary data sets
- ▶ zFS data sets (they are VSAM)

#### How this is addressed in z/OS 1.11

z/OS 1.11 delivered support for all extended format sequential data sets to be EAS-eligible.

#### How this is addressed in z/OS 1.12

In z/OS 1.12 and later, the following data set types are EAS-eligible:

- ▶ BCS and VVDS catalog data sets
- ▶ Sequential data sets, including extended, basic, and large formats
- ▶ PDS and PDSE data sets
- ▶ Direct (BDAM) data sets
- ▶ Data sets allocated with undefined DSORGs

## Restrictions or considerations

Non-EAS eligible data sets are those that can be allocated only in the track-managed space of an EAV volume. For z/OS 1.12, the non-EAS eligible data sets include:

- ▶ HFS data sets
- ▶ Page data sets
- ▶ VTOC and VTOC index data sets
- ▶ VSAM data sets with embed or keyrange attributes that might have been inherited from prior physical migrations or copies

**Attention:** On a z/OS V1R12 system, the system no longer issues the IEC144I 313-0C open abend because data sets of all data set types are EAS-eligible.

Coexistence requirements:

- ▶ Systems before z/OS 1.12 will not be able to use the EAV enhancements that are described in this section.
- ▶ Systems before z/OS 1.11 will not be able to use EAV enhancements that are described in Table 1 in the IBM Knowledge Center. For more information, see the IBM Knowledge Center at:  
[http://www-01.ibm.com/support/knowledgecenter/SSLTBW\\_1.12.0/com.ibm.zos.r12.idak100/eavr11.htm%23eavr11](http://www-01.ibm.com/support/knowledgecenter/SSLTBW_1.12.0/com.ibm.zos.r12.idak100/eavr11.htm%23eavr11)
- ▶ Release z/OS 1.10 provides toleration of z/OS 1.11 EAV enhancements.
- ▶ Releases before z/OS 1.10 also provide toleration of EAVs at the z/OS 1.10 level.

For more information about how to set up and use EAV, see *z/OS DFSMS Using Data Sets*, SC23-6855 and *DFSMS V1.10 and EAV Technical Guide*, SG24-7617

### 3.4.2 Base Control Program Internal Interface (BCPii)

With z/OS 1.11, IBM provided support that allows authorized applications to query, change, and run operational procedures against the installed System z hardware base through a set of application program interfaces. These applications can access the System z hardware that they are running on and extend their reach to other System z processors within the attached process control (that is, Hardware Management Console) network.

Using the Base Control Program internal interface (BCPii), an authorized z/OS application can perform the following actions:

- ▶ Obtain the System z topology of the current interconnected central processor complexes (CPCs) as well as the images, capacity records, activation profiles, and user-defined image groups that are defined on a particular CPC.
- ▶ Query CPC, image (that is, LPAR), capacity record, activation profile, and user-defined image group information.
- ▶ Set various configuration values related to CPC, image, and activation profiles.
- ▶ Issue commands against CPCs, LPARs, and user-defined image groups to run minor or even significant hardware- and software-related functions.
- ▶ Listen for various hardware and software events that might take place on various CPCs and images throughout the HMC-connected network.

Communication to the Support Element (SE) and Hardware Management Console (HMC) using BCPii is done completely within the base operating system. It therefore does not require communication on an IP network (intranet) for connectivity. This configuration provides complete isolation of your System z hardware communication from any other network traffic within the intranet and internet.

To use BCPii, you must have the appropriate hardware and software environments. Any user of BCPii must have adequate authorization through your security product to access BCPii-defined class profiles. The BCPii interface must be enabled on the hardware side with the appropriate security authorizations in place. For more information about how to set up and use BCPii, see Chapter 4, “z/OS enhancements: Base Control Program” on page 57.

The name of the cataloged procedure that IBM supplies in SYS1.PROCLIB for starting the BCPii address space is HWISTART. The STC name is HWIBCPii.

Use the **STOP HWIBCPii** command to shut down the BCPii address space. You might choose to issue this command to stop BCPii for maintenance updates or repeated failures of BCPii services. The system stops the HWIBCPii address space and BCPii interfaces are no longer available.

Use the **START HWISTART** command to restart the BCPii address space after the operator stops or cancels the HWIBCPii address space, or after the address space stops on its own. The system restarts the HWIBCPii address space and all BCPii interfaces become available. The system START command then issues event notification facility (ENF) signal 68 to signal that BCPii APIs can now be issued.

Current BCPii user components are System Automation for z/OS and System Status Detection (SSD) partitioning protocol. In a shop, the HMCs are normally connected so users of BCPii can see all processors, see their status (used by SSD), and issue commands (used by GDPS through **SA-ZOS: IPL LOAD** command for an image).

## 3.5 Diagnostics for availability

This section describes a number of features, runtime diagnostics, and commands that contribute to availability.

### 3.5.1 IBM Health Checker for z/OS

The IBM Health Checker for z/OS function has its origins in helping availability. It was created to help you find potential configuration problems before they can affect you. When thinking about it that way, all IBM Health Checker for z/OS checks can help you diagnose a possible availability problem.

Since z/OS V1.8, over 114 availability health checks have been incorporated, and more are added all the time. It does pay to run IBM Health Checker and review the check messages. Although your installation might not agree with what the check is verifying, give due process to each check because it probably represents a problem someone encountered, and the component owner thought it important enough to have the community-at-large verify the setting.

Some Independent Software Vendors have embraced IBM Health Checker as well by implementing checks that they feel are key within their products.

Although not all 114 newer checks are mentioned here, the ones that are mentioned in this book are cross-referenced into Table 3-2. It shows the name of the health check, the reason for the check, and the z/OS version and release level in which it was introduced, along with a reference to its description in this book. For a complete description of all z/OS-provided health checks, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

Table 3-2 Health checks described in this book

Name	Reason	z/OS level
<b>USS_PARMLIB</b> 3.2.2, "Syntax checker: z/OS UNIX" on page 28	Determines whether there are differences between the current system settings and the settings defined in the BPXPRMxx member.	1.9
<b>SLIP_PER</b> 3.5.4, "Enhanced SLIP DUMP for non-dispatchable (MAXNDSP and DEFERTND)" on page 42	Sees whether a SLIP PER trap has been continuously active for longer than a certain threshold.	2.1
<b>PDSE_SMSPDSE1</b> 3.5.8, "Restartable SMSPDSE1 address space" on page 46	Sees if the SMSPDSE1 address is active to prevent possible PDSE related problems.	1.6
<b>XCF_CDS_SPOF</b> 3.5.13, "Detecting a single point of failure programmatically (IOSSPOF)" on page 51	Checks for appropriate separation of different types of couple data sets, and checks XCF signaling paths and structure sizes.	1.10
<b>XCF_CDS_SEPARATION</b> 3.5.13, "Detecting a single point of failure programmatically (IOSSPOF)" on page 51	Checks that Sysplex couple data set and function couple data sets are properly isolated.	1.4
<b>ALLOC_ALLC_OFFLN_POLICY</b> 4.2, "Change nearly any value in ALLOCxx without an IPL" on page 59	Checks the setting of the ALLC_OFFLN_POLICY parameter.	1.13
<b>ALLOC_SPEC_WAIT_POLICY</b> 4.2, "Change nearly any value in ALLOCxx without an IPL" on page 59	Checks the setting of the SPEC_WAIT POLICY parameter	1.13
<b>ALLOC_TIOT_SIZE</b> 4.2, "Change nearly any value in ALLOCxx without an IPL" on page 59	Checks the setting of the TIOT SIZE parameter ASM_PLPA_COMMON_SIZE	1.13
<b>ASM_PLPA_COMMON_SIZE</b> 4.3, "More effective use of space in ECSA" on page 60	Checks the combined size of the PLPA and Common page data sets in relation to the size of CSA/ECSA and PLPA/EPLPA	V1.8
<b>CSVTAM_CSM_STG_LIMIT</b> 4.3, "More effective use of space in ECSA" on page 60	Checks if the maximum number of bytes of fixed storage dedicated to CSM use and the maximum amount of storage dedicated to ECSA CSM buffers is adequate to meet the needs of your system	1.8
<b>VSM_CSA_LARGEST_FREE</b> 4.3, "More effective use of space in ECSA" on page 60	Monitors the current size of the largest contiguous free block of CSA/ECSA against the installation-specified or default minimum value	2.1
<b>VSM_CSA_THRESHOLD</b> 4.3, "More effective use of space in ECSA" on page 60	Checks the current allocation of CSA and ECSA storage.	1.4
<b>RSM_HVSHARE</b> 4.3, "More effective use of space in ECSA" on page 60	Checks the configured size and current allocation of the high virtual shared area (HVSHARE in IEASYSxx).	1.5

Name	Reason	z/OS level
<b>RSM_MEMLIMIT</b> 4.3, "More effective use of space in ECSA" on page 60	SMFPRMxx, which affects the amount of high virtual storage available to jobs on the system.	1.4
<b>CSV_LNKLST_NEWEXTENTS</b> 4.4, "PROG enhancement: Dynamic LNKLST" on page 62	Sees whether the number of extents in each data set of a LNKLST set has changed since the LNKLST was activated. All active LNKLST sets are checked.	1.4
<b>CSV_LNKLST_SPACE</b> 4.4, "PROG enhancement: Dynamic LNKLST" on page 62	Determines whether any active LNKLST sets on the system were created with secondary space defined for PDSs.	1.4
<b>CSV_LPA_CHANGES</b> 4.5, "PROG enhancement: Dynamic LPA" on page 64	Compare the current IPL's LPA to the previous IPL's LPA.	1.9
<b>SUP_SYSTEM_SYMBOL_TABLE_SIZE</b> 4.10, "Dynamically change system symbols" on page 70	Helps you determine whether you have reached a specified threshold in the system symbol table size.	2.1
<b>VLF_MAXVIRT</b> 4.19, "New MODIFY VLF command" on page 83	Displays the VLF Class and MaxVirt. Turn on VERBOSE option for more information.	2.1
<b>CNZ_EMCS_Inactive_Consoles</b> 4.20, "Add/Remove MCS consoles dynamically" on page 84	Ensures that there are not an excessive number of inactive EMCS consoles	1.4
<b>CNZ_Console_Operating_Mode</b> 4.20, "Add/Remove MCS consoles dynamically" on page 84	Identifies installations that have not explicitly identified their console service operating mode.	1.13
<b>CNZ_OBSOLETE_MSGFLD_AUTOMATION</b> 4.21, "New Message Flood Automation" on page 85	Sees whether an obsolete version of Message Flood Automation is installed.	1.11
<b>SVA_AUTOIPL_DEFINED</b> 4.23, "Auto-IPL using DIAGxx" on page 88	Checks if the customer environment can support an AutoIPL policy, and if it is, determines whether the AutoIPL policy is active.	1.11
<b>SVA_AUTOIPL_DEV_VALIDATION</b> 4.23, "Auto-IPL using DIAGxx" on page 88	Performs device validation for devices specified in the AutoIPL policy for SADMP and MVS when an AutoIPL policy exists.	1.11
<b>IEA_ASIDS</b> 4.24, "z/OS reusable address space support and usage" on page 89	Helps identify how many ordinary and reusable ASIDs are being used. Specify the VERBOSE option to see more information about individual connections to non-reusable ASIDs.	1.9
<b>IEA_LXS</b> 4.25, "z/OS report on linkage indexes" on page 93	Provides a report on the use of system and non-system LXs, both normal and extended. Turn VERBOSE on to get more information about each LX: the LX value, the ASID, and jobname.	1.9
<b>RRS_Storage_NumServerReqs</b> 6.1, "RRS internal restart" on page 100	Monitors the level of virtual storage usage in the RRS address space to prevent a terminating failure	1.7
<b>ICSFMIG7731_ICSF_RETAINED_RSAKEY</b> 12.1, "ICSF: Improve product stability" on page 146	Detects the existence of retained RSA private keys on a PCICC or PCIXCC/CEX2C cryptographic card.	1.9
<b>ICSFMIG7731_ICSF_PKDS_TO_4096BIT</b> 12.1, "ICSF: Improve product stability" on page 146	Verifies that the PKDS size in an ICSF pre-HCR7750 environment is sufficiently allocated to support 4096-bit RSA keys.	1.8

Name	Reason	z/OS level
<b>ICSF_COPROCESSOR_STATE_NEGCHANGE</b> 12.1, "ICSF: Improve product stability" on page 146	Monitors the state of the coprocessors and accelerators daily to detect a negative change in state	2.1
<b>ICSF_MASTER_KEY_CONSISTENCY</b> 12.1, "ICSF: Improve product stability" on page 146	Evaluates the master key states of the coprocessors to detect potential master key problems.	2.1

## Handling the IBM Health Checker for z/OS and checks

Using IBM Health Checker for z/OS's strong customization features is something that you should take advantage of to get full benefit from all the health checks.

Each health check message (either from IBM or an ISV) has component check owner information that can be used to determine to whom to direct the check output. The severity of the exception can be used to determine how to gauge the risk of the problem. Use the severity of the exception to understand any escalation that might be needed. For example, a high severity exception might identify a high risk that needs immediate escalation.

IBM Health Checker for z/OS messages follow a simple and easy formula: Health checker heading message (HZS), followed by check specific messages. This is helpful for applying automation to direct the messages to the correct people.

An example is:

```
HZS0002E CHECK(IBM CNZ,CNZ_SYSCONS_PD_MODE): 430
CNZHF0010E System console SI is running in Problem Determination mode.
```

IBM Health Checker for z/OS: User's Guide describes several automation methods that can be used, but the outcome is clear: Get the messages to the correct people so that any availability concerns can be promptly addressed.

After you've directed the output of the check to the correct person, what should that person do? Ideally, it is best to use that check to work within your installation. That might mean that you change parameters being verified, the check severity, the interval in which it runs, or any of the other options. There are many ways to customize the check to have it work for you. If you cannot find an option to change the check to make it agreeable to your installation, you can delete or deactivate it. Deleting or deactivating a check would be a last resort because if you can get it to perform the way you want, the check will verify your system without future intervention.

Using logstreams for the IBM Health Checker for z/OS can be helpful for capturing data for later diagnostic evaluation. If you have a sporadic exception, this can be particularly helpful.

Some common options in SDSF for manipulating checks are:

- ▶ H: Deactivate the check until you activate it, or the next IPL
- ▶ R: Run the check now. Use this after a problem is corrected to verify it.
- ▶ D L: Determine more specific check information, especially to understand options to override.
- ▶ L: Show a history of check runs. You must be using logstreams to see the history of check runs.



**Tip:** Consider deactivating a check, rather than deleting it. To undelete a check, you must issue the **MODIFY HZSPROC,ADDNEW** command. However, the ADDNEW option will affect all checks on the system. Therefore, if you want run the check in the future, deactivating might be more desirable than a deletion as it might not be unintentionally added back.

### 3.5.2 Runtime Diagnostics component

This section covers concerns about the Runtime Diagnostics component.

#### The issue

Clients are faced with multiple challenges:

- ▶ Availability requirements are increasing.
- ▶ The complexity of the operating environment and the number of products that must be managed is increasing.
- ▶ The number of staff people with many years of problem diagnosis skills is decreasing.
- ▶ The resiliency of key products is improving, meaning that in some instances they are able to run in a degraded mode during a problem rather than abending. These are known as soft failures.

Soft failures are good because they might help you maintain a service to your users. However, because there is no abend, detecting these situations is not always easy. This can result in a slowly degrading user service that can take a long time to detect and diagnose. If the situation is not detected and addressed quickly, the corrective action might be an IPL. In some cases, all that you see is a slowing system. Rather than spend time trying to determine the root cause, you might decide to IPL the system in the hope that will make the problem go away.

#### How this is addressed in z/OS 1.12

Runtime Diagnostics is a base component of z/OS that is designed to help you analyze a system that has potential soft failures. A soft failure (“sick but not dead”) is defined as the combination of typical and abnormal behavior that causes the system to deliver a degraded service.

Runtime Diagnostics runs many of the same tasks you might typically perform when looking for a failure:

- ▶ Reviewing critical messages in the log
- ▶ Examining address spaces with high central processing unit (CPU) usage
- ▶ Looking for an address space that might be in a loop
- ▶ Evaluating local lock conditions

To analyze the home system, Runtime Diagnostics (called HZR) must be started:

**S HZR,SUB=MSTR,OPTIONS=ANALYSIS**

Typically, when analyzing a system, the analysis process takes less than one minute.

Runtime Diagnostics output is sent to the destination specified on the HZROUT DD card. The supplied sample HZR procedure assigns HZROUT to DD DUMMY. To view the HZROUT contents without stopping HZR, you must specify DISP=SHR in the HZR JCL.



Starting with z/OS 1.13, it still runs as a started task, but it remains active until you stop it using the STOP command.

**S HZR,SUB=MSTR** to start Runtime Diagnostics

**P HZR** to stop it

If you want Runtime Diagnostics to analyze the system it is running on (the home system), you initiate the analysis using the MODIFY command:

**F HZR,ANALYZE**

If you want Runtime Diagnostic to run analysis on a different system, you again use the MODIFY command, and include the SYSNAME parameter:

**F HZR,ANALYZE,SYSNAME=SYSB**

When you upgrade to z/OS 1.13 or later, ensure that the HZR JCL contains the correct program name. In z/OS 1.12, it is HZRIMAIN. In z/OS 1.13, it is HZRINIT.

In addition to changing Runtime Diagnostics to not end until you explicitly stop it, z/OS 1.13 also includes other enhancements, such as checking for the following symptoms:

- ▶ ENQ contention checking
- ▶ Local lock suspension
- ▶ z/OS UNIX latch contention
- ▶ GRS latch contention

Also, messages were added (IXC101I, IXC105I, IXC418I) to Runtime Diagnostics analysis

Sample reports and the actions required are documented in the section “Understanding Runtime Diagnostics output” in *z/OS Problem Management*, SC23-6844.

### Restrictions or considerations

Runtime Diagnostics can only run on a z/OS 1.12 or later system. However, it *can* analyze OPERLOG for z/OS 1.11 systems. In that case, you run it on a z/OS 1.12 or later system and specify the **SYSNAME=target** parameter.

The Runtime Diagnostics started task *must* be called HZR. If you use a different name, it will not start successfully. Also, the Predictive Failure Analysis component (PFA) issues commands to Runtime Diagnostics to run analysis on its behalf, and it depends on Runtime Diagnostics being called HZR.

### Rollback

This function is not rolled back to previous releases.

### Further information

For more information about Runtime Diagnostics, see *z/OS Problem Management*, SC23-6844.

## 3.5.3 Runtime Diagnostics with OMVS latch contention

This section details enhancements to Runtime Diagnostics with OMVS latch contention.

### The issue

Before z/OS 1.13, Runtime Diagnostics supported Operlog but not anything in z/OS UNIX.

### How this is addressed in z/OS 1.13

In z/OS V1R13, Runtime Diagnostics (HZR started task) supports analysis of z/OS UNIX latch contention. When a latch contention or waiting threads situation exists for greater than five minutes, the z/OS UNIX file system latch contention and waiting threads record displays at the console.

Runtime Diagnostics gather information about the z/OS UNIX file systems contention. If all the counts are zero, the record is not displayed. Follow the instructions listed in the ACTION statement in the event record by issuing **D OMVS,W,A**. It returns the ASID and job names of any latch waiters.

### Restrictions or considerations

There are no known restrictions related to the use of this function.

### Rollback

This function is not available in the previous release.

### Further information

For more information about Runtime Diagnostics, see *z/OS Problem Management*, SC23-6844.

For more information about OMVS latch contention, see *UNIX System Services Planning*, GA32-0884.

For more information about z/FS File System contention, see *z/OS Distributed File System zSeries File System Administration*, SC23-6887.

## 3.5.4 Enhanced SLIP DUMP for non-dispatchable (MAXNDSP and DEFERTND)

This section details enhancements to the Enhanced SLIP DUMP.

### The issue

With a SLIP, decide whether SDUMP is to quiesce the system (set it to non-dispatchable (Q=YES) or leave the system dispatchable (Q=NO)) while dumping the contents of the SQA and CSA, and collecting Global Exit data. The larger the control blocks and other global storage required, the longer the task is non-dispatchable for program work.

### How this is addressed in z/OS 1.11

In z/OS 1.11, add **MAXNDSP** to CHNGDUMP process to specify the maximum time interval that an SVC dump keeps a system non-dispatchable, where sss is the number of seconds. The default value is 15 seconds.

### How this is addressed in z/OS 1.12

In z/OS 1.12, add **DEFERTND** to CHNGDUMP process to specify whether SDUMP processing should defer setting the tasks of the address space non-dispatchable until the capture of global storage is completed. The default is NO.

For display command:

**D DUMP,OPTIONS**

## Restrictions or considerations

As LPARs get more real storage and programs used above the bar storage or data space, the size of the DUMP SPACE needs to be reviewed.

### **Monitor usage of SLIP with Health Checker:**

Checks to see whether a PER trap has been continuously active for longer than the threshold.

IBMSLIP,SLIP\_PER

## Rollback

This function is not available in the previous release.

## Further information

For more information about the use of the **CHNGDUMP** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 3.5.5 Verify processor type and number with CONFIG

This section describes enhancements to verifying processor type and number with CONFIG.

### **The issue**

Because specialty processors are available, you must identify the number of processors of each type, which are online, and how many are reserved. This information is helpful, for example, when a processor is upgraded or a model is changed. Before z/OS 2.1, the CP could not be identified as a specialty processor, so the **CONFIG** comparison was only the number but not the type.

### **How this is addressed in z/OS 2.1**

In z/OS V2.1, the **DISPLAY MATRIX=CONFIG** and **CONFIG CPU** commands allow you to verify the type (standard CP, zAAP, zIIP) of active processors in a configuration and the number of each.

The **D M** command displays information about your system configuration. The **D M=CPU** gives the list of CPs defined in the image profile, as well as the online/offline status and type of each.

The **D M=CONFIG(xx)** causes the system to display the differences between the current configuration and the configuration described in the **CONFIGxx** PARMLIB member. If you omit xx, the system assumes that you mean CONFIG00.

For a description of the display format, see message **IEE097I**. Use LookAt or use the MVS System Messages books, which can be found under the z/OS MVS topic here:

<http://pic.dhe.ibm.com/infocenter/zos/v2r1/index.jsp>

Example 3-2 shows two options:

1. If CPU 03 is actually offline, but is specified as online in CONFIGxx
2. If CPU 04 is actually online and a standard (general purpose) processor, but is specified as offline and a zIIP processor in CONFIGxx, the message is:

*Example 3-2 Result of D M=CONFIGxx command*

---

CPU	DESIRED	ACTUAL
03	OFFLINE	ONLINE
04	OFFLINE	ONLINE
..	zIIP	STANDARD

---

## Restrictions or considerations

This requires the Hardware Group to update the **CONFIGxx** member per image whenever they do an IODF gen for CHP and update the image profile for CPUs.

## Rollback

This function is not available in the previous release.

## Further information

For more information about the use of the **Display Matrix** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the **CONFIGxx** member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 3.5.6 Diagnosing catalog contention issues

This section describes enhancements for diagnosing catalog contention issues.

### The issue

Catalog contention issues occur and trying to determine the cause can be difficult, leading in some cases to IPL the system to eliminate the contention.

### How this is addressed in z/OS 1.12

In z/OS 1.12, the CATALOG address space Contention Management function monitors the CATALOG address space for possible resource contention. If the wait threshold is exceeded, a message (IEC393I) is issued to the console and a symptom record is written to LOGREC. In addition, a new CONTENTION option on the **MODIFY CATALOG** command was added, as follows:

**F CATALOG,CONTENTION(SYYZTIOT,wait\_time)**

The **F CATALOG,REPORT** shows general information about the catalog address space. The report shows the connection time.

**Important:** If you override the default of 10 minutes and do not update the IGGCATxx member, the command will need to be redone after each IPL. It will still be maintained if the Catalog Address Space (CAS) is recycled.

If resource contention exists for more than that time, message IEC393I is issued, displaying information about the waiting task.

### How this is addressed in z/OS 2.1

In z/OS V2.1, the **F CATALOG,CONTENTION** command shows the resource contention for SYSZTIOT, SYSIGGV2, and SYSZVVDS resources, and for the CATALOG allocation lock.

A new action, REDRIVE, can be associated with a resource and triggered when the wait threshold is breached.

The contention wait-time threshold and the actions to be taken for each can be controlled by using the IGGCATxx PARMLIB member.

To display Contention value, issue this command:

**F CATALOG,REPORT**

### Restrictions or considerations

Receiving IEC393I does not mean that an error or problem exists, so you might not need to take any action. If you cancel any of the jobs listed, first take a memory dump of the **CAS** by issuing this command:

**F CATLOG,TAKEDUMP**

To get more information about the job, issue this command:

**F CATALOG,LISTJ(jobname)**

### Rollback

This function is not available in the previous release.

### Further information

For more information about the use of the IGGCATxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the **MODIFY CATALOG** command, see the section titled “MODIFY CATALOG Command Syntax” in *z/OS DFSMS Managing Catalogs*, SC26-7409 or *z/OS V1.12 DFSMS Technical Update*, SG24-7895.

## 3.5.7 Enhanced D GRS,ANALYZE command

This section covers enhancements to the **D GRS,ANALYZE** command.

### The issue

Your system or sysplex has dropped to idle and there is no apparent reason for it. You are in the process of making an RNL change, but are having difficulties getting the command to work. A batch job is waiting to run but the operator missed the message about contention. A quick check of GRS shows a resource in use. **D GRS,CONTENTION** can be used to identify that resource.

### How this is addressed in z/OS 1.8

The addition of the **D GRS,ANALYZE** command allows you to further identify the holder of the resource. **D GRS,ANALYZE,WAITER** provides a list of tasks that have been waiting the longest. **D GRS,ANALYZE,BLOCKER** shows tasks blocking ENQ resources. **D GRS,ANALYZE,DEPENDENCY** provides a list of tasks with the longest ENQ waiters and the top blockers.

### How this is addressed in z/OS 1.11

Enhanced the Contention Analysis display commands for latches: **D GRS,ANALYZE,LATCH**.

### How this is addressed in z/OS 1.12

Added analysis of z/OS UNIX System Services latches by using the **D GRS,ANALYZE,LATCH** command.

### Restrictions or considerations

The commands are general purpose displays.

### Rollback

This function is not rolled back to previous releases.

### Further information

For more information about the command, see *z/OS MVS System Commands*, SA38-0666.

A more detailed description of **ANALYZE** can be found in *z/OS MVS Setting Up a Sysplex*, SA23-1399.

For more information about the z/OS UNIX latch contention, see *z/OS MVS Diagnosis: Reference*, SA32-0904.

## 3.5.8 Restartable SMSPDSE1 address space

In *z/OS Planned Outage Avoidance Checklist*, SG24-7328, the role of the SMSPDSE and the optional, restartable, SMSPDSE1 address spaces was explained. Today, partitioned data set extended (PDSE) data sets are being used by z/OS, subsystems, ISV products, and client applications. For a better understanding of the use and management of PDSEs, see *Partitioned Data Set Extended Usage Guide*, SG24-6106.

**Important:** It is critical that you adhere to the following PDSE sharing rules to avoid data corruption:

- ▶ If your IGDSMSxx member specifies PDSESHARING(NORMAL), do not share the PDSE data sets beyond the scope of the GRS complex.
- ▶ If your IGDSMSxx member specifies PDSESHARING(EXTENDED), do not share the PDSE data sets beyond the scope of the sysplex.

To enable the SMSPDSE1 address space, the following conditions must exist:

- ▶ The IGDSMSxx initialization parameter, PDSESHARING, must be set to EXTENDED
- ▶ The IGDSMSxx initialization parameters in a sysplex environment must be set as follows:

```
PDSESHARING(EXTENDED)
PDSE_RESTARTABLE_AS(YES)
```

### The issue

Before z/OS 1.6, or if the restartable SMSPDSE1 is not implemented, it might be necessary to IPL one or more systems again to resolve a hang condition, deadlock condition, or out-of-storage condition in the SMSPDSE address space.

Starting with z/OS 1.6, a new restartable address space, SMSPDSE1, was introduced. SMSPDSE1 provides connections to, and processes requests for, those PDSE data sets that



are not part of the global connections that are associated with SMSPDSE. Global connections are those made to PDSE data sets that are in the LINKLST concatenation.

Enable the SMSPDSE1 address space by specifying RESTARTABLE\_PDSE(YES) in your IGDSMSxx member. However, the default (specified in PARMLIB member IGDSMSxx), continues to be to have just a single address space (SMSPDSE).

### How this is addressed in z/OS 1.7 with z/OS Health Checker

A new health checker check (IBMPDSE,PDSE\_SMSPDSE1) was created to check that SMSPDSE1 has started.

The command **D SMS,OPTIONS** shows the options that were read from the IGDSMSxx member. To verify that SMSPDSE1 is started, issue a **D A,SMSPDSE1** command.

For information about the setup and use of SMSPDSE1, see *z/OS DFSMS Using Data Sets*, SC23-6855.

To analyze and repair PDSE and PDSE1 problems, see *z/OS DFSMSdfp Diagnosis*, GY27-7618. Two commands assist with problem determination if problems are encountered with any PDSEs. In the commands, use PDSE to indicate that you want information about SMSPDSE, and PDSE1 for SMSPDSE1.

Use the ANALYSIS keyword on the **VARY SMS** command to help identify the cause of a suspected problem. For example:

```
VARY SMS,PDSE1,ANALYSIS
```

If you determine that the problem is latch contention, use the **FREELATCH** command. This example frees the latch at the specified latch address:

```
VARY SMS,PDSE1,FREELATCH(latchaddr)
```

### Restrictions or considerations

An IPL is required to turn SMSPDSE1 on or off. If you issue a **SET SMS=xx** command pointing at an IGDSMSxx member that contains PDSE\_RESTARTABLE\_AS(YES), the command will appear to complete successfully. If you issue a **D SMS,OPTIONS** command, the response reflects the setting in your IGDSMSxx member. However, the SMSPDSE1 address space will not be started. Similarly, if you IPL the system with PDSE\_RESTARTABLE\_AS(YES) and change the IGDSMSxx member to PDSE\_RESTARTABLE\_AS(NO) and issue the **SET SMS=xx** command, the command appears to end successfully, but the SMSPDSE1 address space will *not* be stopped.

### Rollback

The SMSPDSE1 health check function was rolled back to z/OS 1.6.

### Further information

For more information about SMSPDSE1 setup in DFSMS, see *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

For more information about the IGDSMSxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about defining and starting SMSPDSE1, see *z/OS DFSMS Using Data Sets*, SC23-6855.

For more information about investigating suspected problems in PDSE and restarting the SMSPDSE1 address space, see *z/OS DFSMSdfp Diagnosis*, GY27-7618.

### 3.5.9 Improved IPL-time support for corrupted LNKST PDSE data sets

IPL-time support for corrupted LNKST PDSE data sets has been improved.

#### The issue

In previous releases, when a corrupted PDSE was detected in the linklist during IPL, the system immediately entered a wait state. The wait state code indicates that the cause was a corrupted PDSE. However, the only way to identify the specific PDSE that caused the wait state was to take a stand-alone memory dump and use that to identify the bad PDSE. This can be a time-consuming process.

#### How this is addressed in z/OS 1.12

When a PDSE that is in LNKST is opened during the NIP, the directory structure will be validated. If a corrupted PDSE is encountered, a new message (IGW037I) is issued, identifying the corrupted PDSE data set. The IPL then continues without placing that PDSE into the linklist. The damaged PDSE can then be restored and dynamically added back into LNKST if not required by critical subsystems in early IPL.

#### Restrictions or considerations

The IPL might continue without the data set, but the system might be compromised and might enter a disable wait state of 064 if the corrupted PDSE generates a program check. To avoid that possibility, you can attempt to restore a valid copy of the corrupted data set and IPL the system again.

#### Rollback

This function is not rolled back to previous releases.

#### Further information

The return codes and reason codes contained in the IGW037I message can be found in *z/OS DFSMSdfp Diagnosis*, GY27-7618.

### 3.5.10 Enhanced commands to identify corrupted PDSE control blocks

The commands to identify corrupted PDSE control blocks have been enhanced.

#### The issue

In z/OS releases before 1.13, certain types of problems with a specific PDSE can only be resolved by restarting the SMSPDSE1 address space because there was no ability to target a specific data set. This caused *all* cached PDSE directories to be flushed, and no new PDSE could be opened until SMSPDSE1 completed reinitializing. Until the restart of SMSPDSE1 was completed, the problem can result in failures of currently running jobs and TSO sessions that were accessing PDSEs.

#### How this is addressed in z/OS 1.13

Starting with z/OS 1.13, it is possible to discard the buffered directory pages for individual PDSEs. When an error occurs for a PDSE, the REFRESH parameter on the **V SMS, PDSE<1>** command allows you to specify the PDSE name. This command ensures that on the next access, the data comes from the actual data set.

Issue the command as follows:

```
V SMS,PDSE1,REFRESH,DSN(datasetname)
```

When an IGWNOTIF macro invocation returns a PDSE in use error, use the new CONNECTIONS parameter on the **D SMS,PDSE<1>** command to discover where the PDSE is being used. Example 3-3 shows the output generated.

*Example 3-3 Message with IGWNOTIF*

---

```
ADR795E (ttt)-mmm(yy), AN UNEXPECTED RETURN CODE (return_code) AND  
      REASON CODE (reason_code) HAS BEEN RECEIVED FROM THE IGWNOTIF MACRO  
      WHILE PROCESSING DATA SET datasetname
```

---

You can use this information to solve problems such as determining which users are blocking access to the data set:

```
D SMS,PDSE<1>,CONNECTIONS,DSN(datasetname)
```

Sample output from the command is shown in Example 3-4.

*Example 3-4 Determining connections for a PDSE data set*

---

```
D SMS,PDSE1,CONNECTIONS,DSN(KYNEF.TEST.PDSE)  
IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS00012  
IGW051I PDSE CONNECTIONS Start of Report(SMSPDSE1) 116  
-----data set name----- -----vsgt-----  
KYNEF.TEST.PDSE                                01-DISTS2-002D17  
--asid-- --name-- --tcbl-- -open-  
  0131  KYNEF   006FF188 Input
```

---

## Restrictions or considerations

There are no known restrictions related to the use of this function.

## Rollback

This function was not rolled back to previous releases.

## Further information

For more information about the use of the **V SMS** or **D SMS** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about diagnosing and addressing PDSE issues, see *z/OS DFSMSdfp Diagnosis*, GY27-7618.

### 3.5.11 IEBPDSE utility to check for damaged PDSEs

A new utility was added that can check for damaged PDSEs.

#### The issue

Before z/OS 1.13, there was no tool to identify errors in a PDSE structure. An IEBCOPY or DSS copy of a corrupted PDSE might end with return code 0, and the output data set would also be corrupted. This meant that your process for creating a new sysres (for example) might result in copying corrupted data sets that would cause problems on a subsequent IPL.

### How this is addressed in z/OS 1.13

z/OS 1.13 and subsequent releases provide a PDSE Validation utility that checks specified PDSEs to verify their structural integrity. The utility provides output in the SYSPRINT data set, optionally including a message for each structure or a map that shows the allocation of pages.

The following JCL is used to start the tool, with SYSLIB naming the data set to be validated. Messages are sent to SYSPRINT.

```
//PDSECHK EXEC PGM=IEBPDSE,PARM=""DUMP"  
//SYSPRINT DD SYSOUT=*  
//SYSLIB DD DSN=pdse.data.set.name,DISP=SHR
```

If errors are detected, the program ends with a nonzero return code, with more severe errors resulting in higher return codes.

This utility can be built into your automation, so that (for example) all PDSEs in LNKLIST can be checked as part of the system shutdown procedure. If a problem is found, the operator can decide whether the IPL should be postponed, or the PROGxx member changed so that the damaged PDSE is left out of LNKLIST until it can be repaired.

Include the **PARM="DUMP"** parameter to get a memory dump that might help diagnose any problems that are detected.

### Restrictions or considerations

This utility does not work with PDS data sets and will get an RC=16.

### Rollback

This function is not rolled back to previous releases.

### Further information

For more information about the use of the IEBPDSE program, see *z/OS DFSMSdfp Utilities*, SC26-7414.

## 3.5.12 Dynamically start and stop VSAM record management trace

You can now start and stop the VSAM record management trace without a restart.

### The issue

Before z/OS 1.12, VSAM record management trace could only be enabled by specifying the TRACE parameter on the DD card for the associated VSAM data set. This meant that trace could not be turned on or off without restarting the job or started task.

### How this is addressed in z/OS 1.12

Starting with z/OS 1.12, VSAM serviceability is enhanced by improving the usability of VSAM record management trace. A new interface is provided to let you enable VSAM record management trace dynamically, *without* making any JCL changes. This means that a VSAM record management trace can be started and stopped without impacting application availability.

This capability is implemented by using a new started task called IDAVDT and a new PARMLIB member called IDAVDTxx. To use it, start the IDAVDT started task and use the **MODIFY** command to control the VSAM record management trace.

Use the VSAM Record Management trace to perform these tasks:

- ▶ Capture data when a problem occurs
- ▶ Capture VSAM Record Management control blocks before an error code is passed back to the caller.

To begin the trace, complete the following steps:

1. Review the generalize trace facility (GTF) reserved common storage buffer size to ensure that the buffer is large enough to hold the resulting trace records. The default is 40 K. Increase it to at least 1024 K.
2. Activate GTF on your system. When you start GTF, specify USR or USRP with an AM01 event identifier, which is x'FF5'.
3. Assuming that the IDAVDT started task has already been started, review the IDAVDTxx parameter for correct options. The IDAVDTxx member will look something like this:

```
VTRACE      DSNNAME(SAMPLE.KSDS.DATA)
             JOBNAME(VRMTRA01)
             HOOK(0,1)
             PARM1(100101000000)
             KEYVALUE(x'C1C2C3C4C5C6')
             END
```

Your IBM service representative should be able to help you specify the appropriate values for the particular problem that you are investigating.

4. Use the **MODIFY IDAVDT** command to set up and enable the trace.
5. When complete, use another **MODIFY IDAVDT** command to stop the trace.
6. Stop the GTF task.

### Restrictions or considerations

There are no known restrictions related to the use of this function.

### Rollback

This function was not rolled back to previous releases.

### Further information

For more information about the IDAVDTxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about using the VSAM record management trace, see the section titled “VSAM record management trace facility (non-RLS access)” in *z/OS DFSMSdfp Diagnosis*, GY27-7618.

## 3.5.13 Detecting a single point of failure programmatically (IOSSP0F)

A new method of detecting a single point of failure has been added.

### The issue

An availability risk exists when a single hardware component failure can cause a system function to stop functioning, or worse, cause the entire system to stop functioning. This situation is well-known to z/OS system programmers, but the problem was difficult to solve. With I/O complexity increasing, it is a burdensome task to try to figure out if a single point of failure exists within your I/O configuration. What was needed was a standard programmatic

function to determine whether critical data sets have common hardware components through all paths to a device.

### How this is addressed in z/OS 1.10

The **IOSSPOF** macro is used to check for I/O configuration redundancy of DASD devices or pairs of DASD devices, and explains up to 16 single points of failure in an I/O configuration.

- ▶ Specifying a DEVN1 and a DEVN2 will request a pair of devices and DSNs that are only for message purposes.
- ▶ Specifying HCMSG=YES will have the IOSSPOF macro automatically issue the health checker message in a check environment. Handle= is for remote checks, which are supported.
- ▶ Specifying SPOFArea will return a control block structure that identifies the single points of failure in a way that can be interpreted by software. They are mapped by IOSDSPOF.

Various checks can be switched off if not necessary.

XCF and XES plan to extend and enhance their existing health checks to provide new and improved checks to detect single points of failure for all types of couple data sets. They use the new IOSSPOF service, check for appropriate separation of different types of couple data sets, and check XCF signaling paths and structure sizes.

IOSPF208I Volume vvvvvv (sdddd) has all CHPIDs share a single point of failure. Component indicators are model-dependent.

The IOSPF208I message uses hardware interrogation to determine single points of failure in the CHPID setup based on shared components like:

- Book
- MBA or Fanout
- Domain
- I/O card

The IOSSPOF service is integrated with Health Checker so messages are standardized for each type of failure. See Example 3-5.

#### *Example 3-5 XCF using IOSSPOF for Health Checker*

---

```
CHECK(IBMxcf,XCF_CDS_SPOF)
START TIME: 03/31/2008 11:28:29.798595
CHECK DATE: 20070730 CHECK SEVERITY: HIGH
```

IOSPF251I Volumes WLMPKP (0485) and WLMPKA (0486) share a logical subsystem.

IOSPF203I Volume WLMPKP (0485) has only one online path

IOSPF253I Volumes LOGPKP (0487) and LOGPKA (0488) share the same physical control unit.

IOSPF209I Volume LOGPKA (0488) has all control unit interfaces share 4 of 4 common components.

IOSPF253I Volumes FDSPKP (0489) and FDSPKA (048A) have all paths share the same switch.

IXCH0907I Describes couple data set configurations.

CDS Type	= The type of couple data set, ex. "CFRM" or "SYSPLEX"
Use	= The word PRI or ALT to indicate the primary or alternate couple data set respectively.
Volser	= The volume serial on which the couple data set resides.
Unit	= The device number of the unit associated with the volume on which the couple data set resides.
Data Set Name	= The couple data set name.

---

### How this is addressed in z/OS 1.11

In z/OS 1.11, the SPOFAREA is obtained by the service and must be released by the caller using the length and subpool specified in the SPOFAREA.

### Restrictions or considerations

IOSSPOF does NOT work on Offline devices.

### XCF check using IOSSPOF with Health Checker.

Checks that Sysplex couple data set and function couple data sets are properly isolated.

XCF\_CDS\_SEPARATION

### Rollback

This function is not available in the previous release.

### Further information

For more information about the IOSSPOF function, see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*, SA22-7610.

## 3.5.14 Hardware Instrumentation Service (HIS) detects change in CPU speed

This can now be done without an IPL.

The Hardware Instrumentation Services (HIS) function provides a framework for collecting and recording hardware event data for IBM System z10® machines. This function collects hardware event data for processors in SMF records type 113, subtype 2, and z/OS UNIX output files. You can only use HIS for IBM System z10 or later systems.

### The issue

A HIS performance run can span across “significant hardware events” that render any data collected useless without knowledge of the event. Lining up of SMF Record Type 113 data with other SMF records is a tedious task. Creating a load module mapping of the system might be useful without associated instrumentation data.

These are significant hardware events (state changes):

- ▶ Replacement Capacity (Customer Initiated Upgrade)
- ▶ On/Off Capacity on demand
- ▶ Cooling problems

## How this is addressed in z/OS 1.12

How HIS reacts depends on the STATECHANGE parameter that is specified at the start of the collection run:

- ▶ STATECHANGE=STOP ? Stop the collection run when the event is detected.
- ▶ STATECHANGE=IGNORE ? Continue the collection run as though the event never happened.
- ▶ STATECHANGE=SAVE ?
  - Default
  - Record the previous state of the system (Save all data). Write and close the .CNT file. Close all .SMP files (1 per CPU). Cut SMF Type 113 Records (1 per CPU).
  - Continue the collection run with the new state. Create .SMP files (1 per CPU). Cut SMF Type 113 Records (1 per CPU).

New format of the output files.

Was: SYSHISyyyymmdd.hhmmss.type.cpu#

Now: SYSHISyyyymmdd.hhmmss.seq.type.cpu#

All files from the same collection run have the same prefix:

SYSHISyyyymmdd.hhmmss

The new sequence number represents a collection run interval. A collection run interval is data during a collection run where the system was in a known consistent state (Example 3-6).

The contents of the .CNT file is an indicator as to when the state change occurred.

---

### *Example 3-6 HIS Collection Run with CoD event*

---

A collection run was started at 09:53:26 on January 2, 2010 on a machine with two CPUs, collecting both counter and sampling data, with STATECHANGE=SAVE. During this collection run, CoD increased the capacity of the machine

```
SYSHIS20090102.095326.000.CTR .  
SYSHIS20090102.095326.000.SMP.00 .  
SYSHIS20090102.095326.000.SMP.01 .  
SYSHIS20090102.095326.001.CTR .  
SYSHIS20090102.095326.001.SMP.00 .  
SYSHIS20090102.095326.001.SMP.01 .  
SYSHIS20090102.095326.001.MAP
```

---

The z/OS 1.10 **DISPLAY HIS** command displays the hardware event data collection status for the hardware instrumentation services (HIS). The system displays HIS information in message HIS015I.

## Restrictions or considerations

There are no known restrictions related to the use of this function.

## Rollback

APAR OA30486 rolled back to z/OS 1.10



## **Further information**

For more information about HIS, see *z/OS Version 2 Release 1 Technical Updates*, SG24-8140.

### **3.5.15 Predictive Failure Analysis (PFA)**

Predictive Failure Analysis (PFA) has been added.

#### **The issue**

Failures occur that can be predicted and prevented with adequate notification. Tools such as performance monitors often are only good after the fact for determining the cause of a system or task failure. Many times, system messages are not seen because busy system consoles are receiving messages at the rate of 1,000 or more per minute.

#### **How this is addressed in z/OS 1.10**

Predictive Failure Analysis is a set of checks that are implemented to monitor and predict trends in your system. If used in your system, PFA uses IBM Health Checker for z/OS to display the information.

The tests that came with PFA in V1.10 included Common Storage usage and LOGREC arrival rate. Extra tests that have since been added.

#### **Restrictions or considerations**

PFA uses z/OS UNIX to store its data. When you configure the PFAUSER user ID, you want to conform to your own installation standards. Its home file system is where it stores the data it collects. It does not clean up, so determine how much data you want to keep. Be careful in a shared file system where the home directory might default to the same on all systems because there should be an instance on each system.

PFA uses Java v1.5 or higher.

If z/OS Health Check is active, you can use the SDSF command CK to display details of the PFA checks. Place your cursor next to the entry and type an “S” for the checks’ report.


#### **Rollback**

PFA was not rolled back into previous releases.

## **Further information**

For more on PFA including configuring the task, see *z/OS Problem Management*, SC23-6844.





## z/OS enhancements: Base Control Program

This chapter provides information about enhancements to the z/OS Base Control Program from Version 1.8 on that help you avoid or postpone a planned outage of z/OS or one of its components.

This chapter includes the following sections:

- ▶ List of Base Control Program enhancements
- ▶ Change nearly any value in ALLOCxx without an IPL
- ▶ More effective use of space in ECSA
- ▶ PROG enhancement: Dynamic LNKST
- ▶ PROG enhancement: Dynamic LPA
- ▶ PROG enhancement: Dynamic exit replacement
- ▶ SETPROG enhancement: SVC enhancement
- ▶ Enhancements to LLA processing
- ▶ System REXX availability enhancements
- ▶ Dynamically change system symbols
- ▶ Add more than RESERVED number of CPs without an IPL
- ▶ Switching between primary and secondary DASD without an IPL
- ▶ Specify NOBUFFS action (SMF) at the log stream level
- ▶ Forcing hung commands to avoid an IPL
- ▶ Dynamically configure storage-class memory (SCM)
- ▶ End a task with new operand on FORCE
- ▶ Auto-reply for WTOR
- ▶ DCCF support for WTOR Auto-Reply
- ▶ New MODIFY VLF command
- ▶ Add/Remove MCS consoles dynamically
- ▶ New Message Flood Automation
- ▶ Added LPA defer wait capability
- ▶ Auto-IPL using DIAGxx
- ▶ z/OS reusable address space support and usage
- ▶ z/OS report on linkage indexes

## 4.1 List of Base Control Program enhancements

This section contains information about enhancements to z/OS that give you more control over when you will IPL your system. Table 4-1 provides a list of those covered in this chapter.

*Table 4-1 Distributed File Service - zFS availability enhancements*

Enhancement	Release	Summary	Extra Material
4.2, "Change nearly any value in ALLOCxx without an IPL" on page 59			
4.3, "More effective use of space in ECSA" on page 60			
PROG enhancements starting on page 62			
4.8, "Enhancements to LLA processing" on page 67			
4.9, "System REXX availability enhancements" on page 68			
4.10, "Dynamically change system symbols" on page 70			
4.11, "Add more than RESERVED number of CPs without an IPL" on page 73			
4.12, "Switching between primary and secondary DASD without an IPL" on page 75			
4.13, "Specify NOBUFFS action (SMF) at the log stream level" on page 76			
4.14, "Forcing hung commands to avoid an IPL" on page 77			
4.15, "Dynamically configure storage-class memory (SCM)" on page 78			
4.16, "End a task with new operand on FORCE" on page 79			
4.17, "Auto-reply for WTOR" on page 80			
4.18, "DCCF support for WTOR Auto-Reply" on page 81			
4.19, "New MODIFY VLF command" on page 83			
4.20, "Add/Remove MCS consoles dynamically" on page 84			
4.21, "New Message Flood Automation" on page 85			
4.22, "Added LPA defer wait capability" on page 87			
4.23, "Auto-IPL using DIAGxx" on page 88			
4.24, "z/OS reusable address space support and usage" on page 89			
4.25, "z/OS report on linkage indexes" on page 93			

## 4.2 Change nearly any value in ALLOCxx without an IPL

Thanks to recent updates, you can now change nearly any value in ALLOCxx without having to do an IPL.

### 4.2.1 The issue

Before z/OS 1.11, changes to any of the settings in the ALLOCxx PARMLIB member required an IPL. There were no commands to display the current options or the default values.

### 4.2.2 How this is addressed in z/OS 1.11

Starting in z/OS 1.11, the new **SETALLOC** command allows the individual system settings that are specified in the ALLOCxx member to be dynamically changed. In support of that capability, the new **DISPLAY ALLOC** command displays all parameters that are in effect.

### 4.2.3 Restrictions or considerations

The installation defaults for handling allocation requests can be overridden by installation exit routines that are specified in the EXITxx PARMLIB member. For example, you can code an exit or use an ISV-supplied one to not cause a production job to be canceled when the VOLUME\_ENQ POLICY setting is CANCEL.

If the **SETALLOC** command is used to make dynamic changes, you must remember to update the ALLOCxx member to avoid regressing those changes at the next IPL.

Now that system settings that are controlled by the ALLOCxx member are dynamic, it is easier to make (and back out) changes to those settings. Therefore, review your current ALLOCxx settings to identify if any changes would improve availability. In particular, investigate what action should be taken when a job that creates a new data set but the data set cannot be cataloged because there is already a catalog entry for that data set (NOT CATLG 2 message). The default is that this situation is not considered to be an error. However, the CATLG\_ERR parameter allows you to specify that a job step that gets a NOT CATLG 2 error fails. This avoids later accidentally processing a back-level data set.

**Tip:** z/OS 1.11 introduced a new ALLOCxx parameter, **SYSTEM IEFBR14\_DELMIGDS(NORECALL)** that eliminates HSM recalls when you delete a migrated data set.

z/OS 1.13 included a number of new health checks that check the setting of some ALLOCxx settings:

- ▶ **ALLOC\_ALLC\_OFFLN\_POLICY** checks the setting of the **ALLC\_OFFLN POLICY** parameter.
- ▶ **ALLOC\_SPEC\_WAIT\_POLICY** checks the setting of the **SPEC\_WAIT POLICY** parameter.
- ▶ **ALLOC\_TIOT\_SIZE** checks the setting of the **TIOT SIZE** parameter.

### 4.2.4 Rollback

This function is not rolled back to previous releases.

## 4.2.5 Further information

For more information about the use of the **SETALLOC** and **DISPLAY ALLOC** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the **ALLOCxx** member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.3 More effective use of space in ECSA

The system now makes more effective use of space in ECSA storage.

### 4.3.1 The issue

With 31-bit addressability, the private region is limited to 2 GB. ECSA storage reduces the private region by the size of ECSA. As more products place their start programs and cross-address space control blocks in ECSA, the amount of above-the-line storage that is used by ECSA can become a limiting factor for application growth. New releases of z/OS and the major subsystems often include enhancements to reduce their use of common storage. However, those savings are often offset by workload growth.

### 4.3.2 How this is addressed in z/OS 1.10

z/OS 1.10 introduced a new virtual storage structure using 64-bit storage. This new area is known as above-the-bar. The area below the 2 GB bar is known as below-the-bar. 64-bit addresses support up to 16 exabytes (16 with 18 zeros after it, or 16,000,000,000,000,000,000 bytes). The above-the-bar storage is divided into three areas:

- ▶ Low non-shared private (2 GB to 2 TB)
- ▶ Shared memory (2TB to 512TB)
- ▶ High non-shared private (512TB to 16 exabyte (2-base-64))

Dataspaces are separate from this 64-bit storage. Each program has the region addressability to all the 64-bit storage, as shown in Figure 4-1

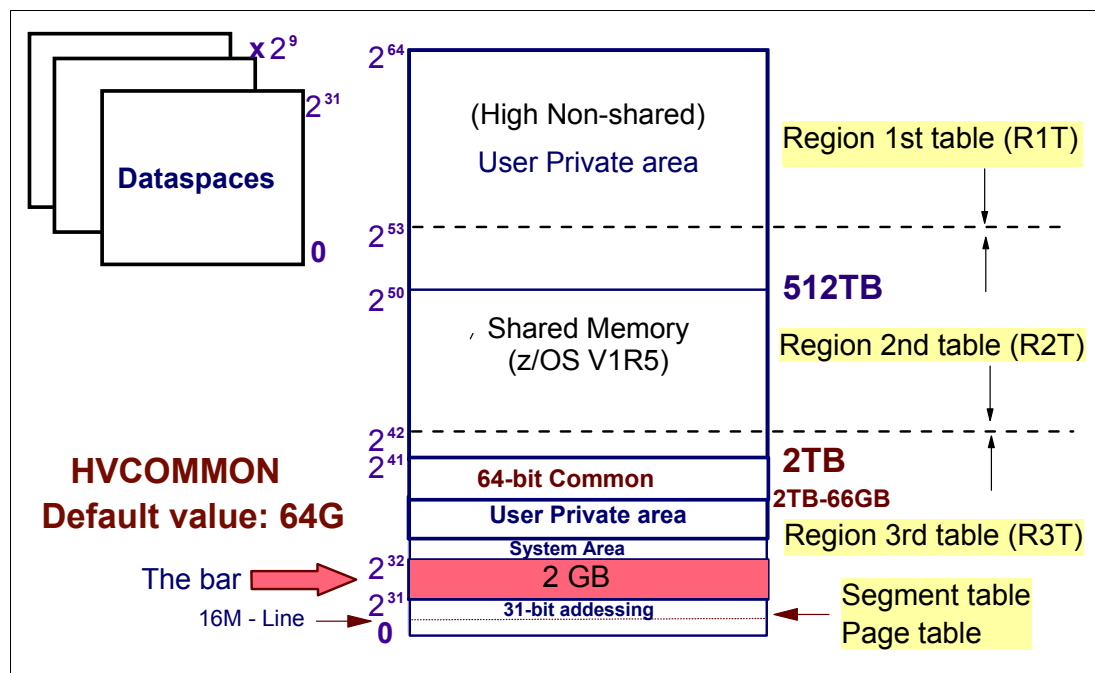


Figure 4-1 Map of 64-bit storage

A new common area called High Virtual Common (HVCCOMMON) is defined in the IEASYSxx PARMLIB member. For larger systems, the default value of 64 GB might not be sufficient. This value can be up to 1 TB plus 2 GB (2 GB is allocated for the operating system) and must be a multiple of 2 GB. It is allocated from the 2 TB line down in the low non-shared private area. To display the current value of this setting, use the following command:

**D VIRTSTOR,HVCCOMMON**

The MVS system resources manager (SRM) monitors how much of the specified 64-bit Common Storage (HVCCOMMON) is in use and generates an alert when the usage exceeds certain thresholds. The following messages get issued by SRM when 80% or 95% of the 64-bit common storage is in use, and when the usage drops back below the threshold:

```
IRA130E HIGH COMMON SHORTAGE
IRA131E CRITICAL HIGH COMMON SHORTAGE
IRA132I HIGH COMMON SHORTAGE RELIEVED
```

### 4.3.3 How this is addressed in z/OS 1.13

Over time, system components will convert to use 64-bit rather than 31-bit storage, freeing up 31-bit storage for application use. Also, if you have a problem with common storage creep, freeing up common storage might allow you to run for longer before you are forced to do an IPL.

In z/OS 1.13, some TCP/IP control blocks are relocated from ECSA to HVCCOMMON and the TCP/IP data space is eliminated. Also, some VTAM control blocks are relocated from ECSA to HVCCOMMON and the VTAM data space is eliminated. With z/OS 1.13, moving the VTAM internal trace (VIT) to HVCCOMMON can make up to 999 pages of ECSA available to other applications or subsystems.

### 4.3.4 Restrictions or considerations

DB2 V10 and later requires storage in both the HVCOMMON and HVSHARE area. See the installation guide for your version of DB2 (for example, *DB2 for z/OS Installation and Migration Guide*) for information about how much storage is required by each DB2 subsystem. Remember to allow for the situation where you move a DB2 subsystem between z/OS systems, in a failover scenario for example.

Various releases of z/OS contain different Health Checker checks that monitor various aspects of common storage usage:

- ▶ The ASM\_PLPA\_COMMON\_SIZE check (delivered in z/OS 1.8) checks the combined size of the PLPA and Common page data sets in relation to the size of CSA/ECSA and PLPA/EPLPA.
- ▶ The CSVTAM\_CSM\_STG\_LIMIT check (delivered in z/OS 1.8) checks whether the maximum number of bytes of fixed storage dedicated to CSM use and the maximum amount of storage dedicated to ECSA CSM buffers are adequate to meet the needs of your system.
- ▶ The VSM\_CSA\_LARGEST\_FREE check (delivered in z/OS 1.12) monitors the current size of the largest contiguous free block of CSA/ECSA against the installation-specified or default minimum value.
- ▶ The VSM\_CSA\_THRESHOLD check (delivered in z/OS 1.4) checks the current allocation of CSA and ECSA storage.
- ▶ The RSM\_HVSHARE check (delivered in z/OS 1.5) checks the configured size and current allocation of the high virtual shared area (HVSHARE in IEASYSxx). This check issues a warning when the allocation of high virtual storage exceeds a predetermined threshold, or when the size of the high virtual shared area is less than the default minimum.
- ▶ The RSM\_MEMLIMIT check (delivered in z/OS 1.4) checks the MEMLIMIT parameter in SMFPRMxx, which affects the amount of high virtual storage available to jobs on the system.

### 4.3.5 Rollback

APAR OA34311 allows subspace direct access to 64-bit private and shared storage (CICS is one user of this capability) and is rolled back to z/OS 1.12.

### 4.3.6 Further information

For more information about the use of the HVCOMMON parameter in the IEASYSxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the 64-bit storage, see *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.4 PROG enhancement: Dynamic LNKLIST

This section describes the enhancements to dynamic LNKLIST.



## 4.4.1 The issue

Before z/OS 1.12, the **COPYFROM** statement had to be specified when you wanted to define a new LNKST set based on an existing LNKST set when using the dynamic LNKST function. If you forgot to specify **COPYFROM**, the result was an empty LNKST set. Upon activating that LNKST set, data sets were missing from LNKST. Missing data sets in the link list can cause application availability problems.

## 4.4.2 How this is addressed in z/OS 1.12

With z/OS 1.12, you can set up multiple default options that reduce errors when using dynamic LNKST functions. This can protect you if you forget to specify a base LNKST set. The defaults that you specify apply to:

- ▶ Statements in the PROGxx member
- ▶ The **SETPROG** command

The defaults do *not* apply to the dynamic LNKST programming interface.

When creating a LNKST set, if you want to require users to always specify a LNKST set that the new LNKST set is to be based on, specify the following in PROGxx:

```
DEFAULTS LNKST REQCOPYFROM
```

By specifying only **REQCOPYFROM**, the **SETPROG** command is rejected if a **COPYFROM** statement is not specified.

If you want to use the current LNKST set as a base when **COPYFROM** was not specified, in PROGxx specify:

```
DEFAULTS LNKST COPYFROMCUR
```

Finally, you can use these two defaults with each other. When you specify **DEFAULTS LNKST REQCOPYFROM COPYFROMCUR**, this means that a base LNKST set is required as a starting point. If a starting LNKST set is not specified, the current LNKST set is used.

To determine which PROGxx DEFAULTS are in effect, use the following command:

**D PROG,DEFAULTS**

In this example shown in Example 4-1, the default has been set to **COPYFROMCUR**. You want to add a data set at the end of the current LNKST and activate the change.

*Example 4-1 Commands with a COPYFROMCUR default*

---

```
LNKST DEFINE NAME(NEWLLSET)
LNKST ADD NAME(NEWLLSET) DSNAME(dataset.to.be.added)
LNKST ACTIVATE NAME(NEWLLSET)
```

---

In another example, the default has been set to **REQCOPYFROM**. You want to add a data set at the end of an existing LNKST set called **OLDLLSET** and activate the change. The statements shown in Example 4-2 are used.

*Example 4-2 Commands with a REQCOPYFROM default*

---

```
LNKST DEFINE NAME(NEWLLSET) COPYFROM(OLDLLSET)
LNKST ADD NAME(NEWLLSET) DSNAME(dataset.to.be.added)
LNKST ACTIVATE NAME(NEWLLSET)
```

---

### 4.4.3 Restrictions or considerations

If the **SETPROG** command is used to implement dynamic options (instead of using the **SET PROG=xx** command), you must update the PROGxx member to avoid regressing options at IPL time.

Use health check CSV\_LNKLST\_NEWEXTENTS to see whether the number of extents in each data set of a LNKLST set has changed since the LNKLST was activated. All active LNKLST sets are checked.

Use health check CSV\_LNKLST\_SPACE to determine whether any active LNKLST sets on the system were created with secondary space defined for PDSs.

### 4.4.4 Rollback

This function is not rolled back to previous releases.

### 4.4.5 Further information

For more background information about dynamic LNKLST, see *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

For more information about the use of the **SETPROG** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the PROGxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 4.5 PROG enhancement: Dynamic LPA

This section describes the enhancements to dynamic LPA.

### 4.5.1 The issue

When using dynamic LPA and adding a module, be careful that you also add all associated aliases. It is a common problem to update a module in LPA and forget to update its aliases. This, in turn, can cause application problems if the “unupdated” alias is referenced after using dynamic LPA.

### 4.5.2 How this is addressed in z/OS 1.12

Starting with z/OS 1.12, this problem can be avoided by using the new **ADDALIAS** option when using dynamic LPA add with either of these commands:

- ▶ The PROGxx PARMLIB member: **LPA ADD MOD(xxx),ADDALIAS**
- ▶ The system command: **SETPROG LPA,ADD,MOD(xxx),ADDALIAS**

Even better, you can make **ADDALIAS** the default for all **LPA ADDs**, by coding it as the default in the PROGxx member by using this command:

```
DEFAULTS LPA ADDALIAS
```

**Tip:** Specifying **DEFAULTS LPA ADDALIAS** is a good system programmer practice to ensure that aliases are not forgotten when adding or updating LPA modules. (IBM is not aware of any situation where it is *not* good to add all aliases when using dynamic LPA add.

### 4.5.3 Restrictions or considerations

Use health check CSV\_LPA\_CHANGES to perform these tasks:

- ▶ Compare the current IPL's LPA to the previous IPL's LPA. You can see information about modules that have changed in size (or been added or removed), along with summaries of the storage deltas for each of the LPA subareas (PLPA, MLPA, FLPA, device support, dynamic LPA).
- ▶ See totals for each of the subareas.

In both cases, the display differentiates between the below-16M area and the above-16M area.

### 4.5.4 Rollback

This function is not rolled back to previous releases.

### 4.5.5 Further information

For more background information about dynamic LPA, see *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

For more information about the use of the **SETPROG** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the PROGxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 4.6 PROG enhancement: Dynamic exit replacement

This section describes the enhancements to dynamic exit replacement.

### 4.6.1 The issue

You can use the z/OS dynamic exit capability to add and delete dynamic exits. If you wanted to replace a routine, there would be a period when the dynamic exit routine was not available (between the delete and the add). Even if that time frame is short, it might still be an exposure to not have the dynamic routine exit routine available for processing.

### 4.6.2 How this is addressed in z/OS 1.12

As of z/OS 1.12, the **SETPROG EXIT** command supports the **REPLACE** command. When replacing an active exit routine, there is always one (and only one) copy of an exit routine that gets control at the exit point. There is no point at which the active exit routine is not available.

A sample command to do this is:

```
SETPROG EXIT,REPLACE,EXITNAME=IRREVX01,MODNAME=IRREVX1A,DSNAME=SY1.EXITS
```

### 4.6.3 Restrictions or considerations

In z/OS 1.12, an enhancement to the PROGxx PARMLIB member allows you to specify the exit type to display by default when using the **DISPLAY PROG,EXIT** command. In PROGxx, the syntax is:

```
DEFAULTS EXIT EXITTYPE(INSTALLATION|NOTPROGRAM|ALL)
```

For example, if you wanted to see only the installation exits on the display command, specify the following in PROGxx: **DEFAULTS EXIT EXITTYPE(INSTALLATION)**.

Issuing a **DISPLAY PROG,EXIT** command might produce a report like that shown in Example 4-3.

*Example 4-3 Sample display using DEFAULTS EXIT EXITTYPE(INSTALLATION) in PROGxx*

---

```
D PROG,EXIT
CSV460I 20.36.14 PROG,EXIT DISPLAY 009
EXIT          DEF EXIT          DEF EXIT          DEF
SYS.IEFACTRT  E  SYSSTC.IEFACTRT  E  SYSSTC.IEFU29  E
IXC_ELEM_RESTART  E  IXC_WORK_RESTART  E  SYSTS0.IEFUSI  E
SYSJES2.IEFUJI  E  SYSJES2.IEFACTRT  E  SYSJES2.IEFUSI  E
SYSSTC.IEFUJI  E  SYSSTC.IEFUSI  E  SYSSTC.IEFUS0  E
SYSSTC.IEFUJP  E  SYSSTC.IEFU85  E  SYSSTC.IEFU84  E
SYSSTC.IEFU83  E  SYS.IEFUAV  E  SYS.IEFUTL  E
SYS.IEFUS0  E  SYS.IEFUJP  E  SYS.IEFUSI  E
SYS.IEFUJV  E  SYS.IEFU85  E  SYS.IEFU84  E
SYS.IEFU83  E  HASP.$EXIT0  E  CSVLLIX1  E
CSVLLIX2      E
```

---

### 4.6.4 Rollback

This function is not rolled back to previous releases.

### 4.6.5 Further information

For more background information about dynamic exits, see *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

For more information about the use of the **SETPROG** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the PROGxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 4.7 SETPROG enhancement: SVC enhancement

The following enhancements have been made to the SETPROG command.

## 4.7.1 The issue

Previously, to replace an SVC, you had to use the **EXIT DEL** and **ADD** commands, then the **Modify INACT** and **MODIFY ACT**.

## 4.7.2 How this is addressed in z/OS 1.12

As of z/OS V1R12, you can use dynamic LPA to update the SVC table and replace the SVC routine, with the **SVCNUMDEC** keyword, for example:

```
SETPROG LPA,ADD,MODNAME=module,DSNAME=dsn,SVCNUMDEC=svcnum
```

## 4.7.3 Restrictions or considerations

Use the **SVCNUMDEC** keyword for updating already-defined SVC table entries.

However, to add an SVC routine, you or the provider of the SVC must write a program that gets control at the CSVDYLPA exit routine, looks for a given routine name, and then issues the appropriate **SVCUPDTE** service call to update the SVC table.

## 4.7.4 Rollback

This function is not rolled back to previous releases.

## 4.7.5 Further information

For more information about adding SVCs dynamically, see *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

For more information about the use of the **SETPROG** command, see *z/OS MVS System Commands*, SA38-0666.

For more background information about adding SVCs dynamically, see the section titled “**SVCUPDTE Macro**” in *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

# 4.8 Enhancements to LLA processing

LLA processing has been enhanced.

## 4.8.1 The issue

Before z/OS 1.12, when a **Modify LLA** command was running, a new **Modify** command would get a “busy” error message and the command would be rejected. If this error was not caught, a refresh of a particular data set would be missed and the change would not get implemented until the next IPL.

Additionally, when the **S LLA,NN=xy** command was issued without **SUB=MSTR**, z/OS would automatically restart LLA with **SUB=MSTR** but without the parameter **NN=xt**. It would then abend with missing member or start with the wrong member.

## 4.8.2 How this is addressed in z/OS 1.12

With z/OS 1.12, up to 255 **MODIFY** commands can be started without getting a “busy” message (a 256th would get “busy”). Requests received while an earlier one is in-process get queued and processed in turn. The requester does not get a “busy” response.

Additionally, processing of the **S LLA,NN=xy** command without **SUB=MSTR** is changed so that the command is treated as though both **SUB=MSTR** and **NN=xy** were specified.

## 4.8.3 Restrictions or considerations

There are no restrictions that IBM is aware of.

## 4.8.4 Rollback

This function is not rolled back to previous releases.

## 4.8.5 Further information

For more information about the **MODIFY LLA** command, see *z/OS MVS System Commands*, SA38-0666.

# 4.9 System REXX availability enhancements

System REXX is a z/OS component that allows REXX execs to be run outside of conventional TSO/E and batch environments. System REXX work requests come from either the AXREXX programming interface, or from the **MODIFY AXR** system command.

System REXX was initially available as a z/OS web deliverable that was installable on z/OS 1.8. In z/OS 1.9, System REXX was incorporated into z/OS. Several enhancements have since been added to System REXX by using APARs. Some of these enhancements have focused on improved flexibility and availability.

## 4.9.1 The issue

Originally, the only data set that System REXX could run execs from was SYS1.SAXREXEC, and this exact data set name was required. IBM included System REXX execs in this data set and any client-written System REXX execs had to be stored in that same data set. This restriction posed a number of challenges that made using System REXX less attractive.

The AXR address space is used by System REXX for its infrastructure. It is started automatically at IPL. Subtasks within the AXR address space process TSO=NO requests. TSO=YES requests are handled in separate address spaces (**AXR01-AXR08**). The System REXX address space (AXR) was not restartable.

## 4.9.2 How this is addressed in z/OS 1.9 and later

After System REXX was made available, IBM provided APAR OA26802 to z/OS 1.9 System REXX. This APAR provided these benefits:

- ▶ The ability to have multiple concatenated System REXX data sets.
- ▶ The ability to stop and restart the System REXX address space (AXR).

By restarting System REXX, you can change the list of System REXX concatenated data sets without an IPL.

- ▶ A command to display information about System REXX.

### Data set concatenation

The AXRxx PARMLIB member contains the list of System REXX data sets. The AXR=xx keyword in IEASYSxx is used to specify one or more AXRxx PARMLIB members.

In addition to the list of System REXX data sets, the AXRxx member also defines:

- ▶ The command prefix to be used to initiate System REXX execs.
- ▶ The user ID under which the System REXX execs are to run.

With this support, it is easy to separate IBM-provided System REXX execs, other vendor execs, and your own execs. If you do not specify SYS1.SAXREXEC in your data set concatenation, SYS1.SAXREXEC is automatically added as the last data set. The number of extents in the concatenation cannot exceed 255, and can be PDSs, PDSEs, or a combination of both. A PDSE counts as a single extent regardless of how many extents it consists of.

### Stopping and restarting System REXX

The System REXX address space (AXR) is not cancellable. For more information about AXR and reusable ASIDs, see 4.24, “z/OS reusable address space support and usage” on page 89. If you must stop the System REXX address spaces, one of these methods can be used:

- ▶ Issue **D JOBS,AXR\*** to display the AXRnn address spaces that are active and individually cancel each AXRnn address space. This method is the least disruptive.
- ▶ Use the **FORCE AXR,ARM** command. This command stops both AXR and all the AXRnn address spaces. Note that using this command stops AXR as well, which runs under the MASTER subsystem, and might not need to be stopped. Using the **FORCE** command can have ramifications for your system, so read about the **FORCE** command in *z/OS MVS System Commands*, SA38-0666, before using it.

Having forced the AXR address space, you can restart it by using the AXRPSTRT proc in SYS1.PROCLIB. On the **START** command, you can specify which AXRxx members should be used, as follows:

```
START AXRPSTRT,AXR=(aa,bb,...)
```

In this command, aa, bb are AXRxx PARMLIB members found in the PARMLIB concatenation.

### Commands associated with System REXX

You can obtain status information about System REXX, or start execution of a System REXX exec, by using the **MODIFY AXR** system command. If you use the command prefix that you defined in the AXRxx member, you can start a System REXX exec using that instead of the **MODIFY AXR** command.

To display the status of System REXX, use the **MODIFY AXR,SYSREXX,STATUS** command, as shown in Example 4-4.

*Example 4-4 Example of displaying System REXX status*

---

```
F AXR,SYSREXX,STATUS
AXR0200I SYSREXX STATUS DISPLAY 970
SYSTEM REXX STARTED AT 10.22.01 ON 08/08/2013
PARMLIB MEMBERS:      AXR00
CPF: REXX$A (SYSPLEX)      AXRUSER: AXRUSER
TIMEINT: 30              TMP: NOT ENABLED
SUBSYSTEM:              AXR
REQUESTS QUEUED:        0  ACCEPTING NEW WORK
REXX WORKER TASKS:      ACTIVE:  0      TOTAL:  4
                        IDLE:    4      MAX:   32
                        ASYNC:   0      SYNC:   0
                        UNTIMED:  0
TSO SERVER SPACES:      ACTIVE:  0      TOTAL:  0
                        IDLE:    0      MAX:    8
                        ASYNC:   0      SYNC:   0
                        UNTIMED:  0
```

---

### 4.9.3 Restrictions or considerations

If you want to change any AXRxx value, including the System REXX data set concatenation, you must stop and restart System REXX, specifying the AXRxx PARMLIB members that you want to use on restart.

### 4.9.4 Rollback

APAR OA26802 is available for z/OS 1.9 and later releases.

### 4.9.5 Further information

For more information about the use of the **MODIFY AXR** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the AXRxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the REXX language, see *z/OS IBM Compiler and Library for REXX on zSeries User's Guide and Reference*, SH19-8160.

For more information about the System REXX execs that are run in a TSO=YES environment with CONSNAME=Name of Issuing Console, see *z/OS MVS Authorized Assembler Services Guide*, SA22-7608.

## 4.10 Dynamically change system symbols

System symbols are a powerful tool that system programmers can use to maintain and manage systems efficiently and that can be used to increase system availability. System symbols allow you to reference many different resources with a single name. Without this



central “management tool”, maintaining and managing a large sysplex environment is more difficult and error-prone.

### 4.10.1 The issue

Although system symbols are powerful, the lack of ability to change them without an IPL limited the extent to which they can be used. *z/OS Planned Outage Avoidance Checklist*, SG24-7328, shows how to use a sample program called IEASYMUP that can be used to update system symbols. However, IEASYMUP is only a sample; clients want an officially supported way of changing system symbols dynamically.

### 4.10.2 How this is addressed in z/OS 2.1

z/OS 2.1 provides the ability to change system symbols in a supported way. There are two ways to change system symbols in z/OS 2.1:

- ▶ By using the **SETLOAD** command
- ▶ Using the IEASYMU2 program

It is important to understand how they interact with each other and with IEASYMUP.

Starting in z/OS 2.1, an option on the **SETLOAD** command can be used: **SETLOAD xx, IEASYM**. This option processes the IEASYM statement in the LOADxx member found in the PARMLIB concatenation (or by indicating where to find the data set containing the LOADxx member with the **SETLOAD xx, IEASYM, DSN=dd, VOLUME=vv**).

When the **SETLOAD xx, IEASYM** command is issued, a new complete system symbol table is created and replaces the existing system symbol table. The old system symbol table remains allocated. Notice that a complete system symbol table is created with each **SETLOAD xx, IEASYM** command. It is better to do fewer **SETLOADs** and have more system symbol updates in each one, than to do many **SETLOADs** with fewer system symbol changes. The system symbol table is only updated on the local system, and changes to the system symbol table are *not* sent to the other systems in the sysplex.

A new ENF signal, ENF 73, is issued to indicate that a new system symbol table has been stored. Components listening to this ENF signal might decide to do various individual actions based on how they use system symbols. When a system table is updated successfully, you will see a message such as:

```
IEE900I SYSTEM SYMBOLS WERE UPDATED FROM LOADMW
```

To display the value of all defined system symbols, use the **D SYMBOLS** command.

To display storage usage by system symbols, use the **D SYMBOLS, SUMMARY** command, as shown in Example 4-5.

#### *Example 4-5 System symbols storage usage*

---

```
IEA994I STATIC SYSTEM SYMBOL INFO  
SYMBOLS DEFINED: 35  
CURRENT TABLE SIZE: 1000 BYTES  
MAX TABLE SIZE: 32512 BYTES
```

---

**Note:** The maximum number of symbols you can have depends on their size. IBM gives you the ability to define 800 symbols of your own, but you can probably define more. You can have as many symbols as you can fit into the symbol table. The symbol table can be up to 32512 bytes long, including a 4-byte header.

### 4.10.3 IEASYMU2 and IEASYMUP usage

As of z/OS 2.1, the sample IEASYMUP has been changed to end with a return code of X'FFF'. If you were using IEASYMUP, rebind it with the updated IEASYMUP object deck in SYS1.SAMPLIB to alert you that this program should no longer be used if it happened to be called. If you have a mix of z/OS 2.1 and pre-2.1 systems in your sysplex, you need two versions of IEASYMUP: The original one that works on the pre-2.1 systems and the new one that does not work.

As of z/OS 2.1, a similar program to IEASYMUP, called IEASYMU2, is provided. You can use this program in the same way that you used IEASYMUP on prior releases.

When you use IEASYMU2 (or IEASYMUP before a rebind), the system symbols are updated directly into the current system symbol table. That is, IEASYMU2 (and IEASYMUP before a rebind) do not create a system symbol table, as **SETLOAD xx, IEASYM** does. Note these behaviors when using IEASYMU2 with **SETLOAD xx, IEASYM**:

- ▶ If you change your system symbols using IEASYMU2 and do not harden those changes into the IEASYMxx member, your changes will be lost the next time you IPL the system.
- ▶ Any **SETLOAD xx, IEASYM** command that is issued after usage of IEASYMU2 results in your IEASYMU2 changes being lost if they were not hardened into your IEASYMxx member. Therefore, when you use **SETLOAD xx, IEASYM**, be careful to make sure that any subsequent IEASYMU2 usages included updating the IEASYMxx members that you are using.
- ▶ Using IEASYMU2 will not cause an ENF 73 to be issued. Any component that needs to know about system symbol updates will not be aware of them.

To display the current values of your system symbols, use the **D SYMBOLS** command. In fact, any time you use the **SETLOAD** command or the IEASYMU2 program, display the system symbol values before and after the change.

### 4.10.4 Restrictions or considerations

In addition to the considerations above for IEASYMU2, the IBM Health Checker for z/OS check SUP\_SYSTEM\_SYMBOL\_TABLE\_SIZE helps you determine whether you have reached a specified threshold in the system symbol table size. You can store at least 800 system symbols in the system symbol table.

### 4.10.5 Rollback

There is no rollback of **SETLOAD xx, IEASYM** or IEASYMU2 support to previous releases.

### 4.10.6 Further information

For more information about the use of the **SETLOAD xx, IEASYM** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.11 Add more than RESERVED number of CPs without an IPL

Thanks to recent enhancements, you can add more than RESERVED number of CPs without running an IPL.

### 4.11.1 The issue

At one time, the activation profile for an LPAR contained a definition of the number of CPs the LPAR would be able to use. If the LPAR needed more than that amount of capacity, the profile would need to be updated, and the LPAR would have to be deactivated and then reactivated, which required an IPL.

Later, to improve flexibility, the activation profile was changed to allow you to specify an INITIAL number of CPs, and an extra number that can be added (defined as RESERVED). When the LPAR was activated, the INITIAL number of CPs would be available to the operating system. However, if more capacity was required, you could issue an MVS **CONFIG** command to configure online up to the RESERVED number of extra CPs. However, when the RESERVED number is reached, the only way to grow beyond that would be to update the activation profile and deactivate and reactivate the LPAR.

### 4.11.2 How this is addressed in z/OS 1.10

z/OS 1.10, together with changes in z10 and later CPCs, added the ability to dynamically change the INITIAL and RESERVED values without deactivating and reactivating the LPAR, and for z/OS to detect the change. This capability in z/OS is controlled by the DYNCPADD parameter in the LOADxx member, and is enabled by default.

With this capability, you can use the MVS **CONFIG** command to vary online up to the new combined total number of CPs that are available for use. For example, if the LPAR is using shared CPs, and there are a total of 38 shared CPs available on the CPC, then the LPAR can use up to that many.

### 4.11.3 How this is addressed in z/OS 2.1

In z/OS 2.1 and later releases, the DYNCPADD parameter in the LOADxx member changed. If ENABLE is selected, control blocks are predefined for the largest possible number of PUs on that model of CPC. Before z/OS 2.1, ENABLED was the default.

Starting with z/OS 2.1, the valid parameters are ENABLED, DISABLED, or nnnn, where nnnn is the number of CPUs can be dynamically added to the image for the life of the IPL. If no value is specified, the default is 16.

For more information about the DYNCPADD keyword and selecting the correct value for your environment, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

To display the current number of active and reserved CPs, use the following command:

```
D M=CPU
```

#### 4.11.4 Restrictions or considerations

With z/OS 2.1, specify nnnn to avoid allocating CP-related storage for CPs that will never be dynamically added for the life of the IPL.

#### Tips

Before being able to configure any additional CPs online, you need to change either the INITIAL or RESERVED number of CPs on the HMC. To do this, select the **Logical Processor Add** function on the HMC, as shown in Figure 4-2.

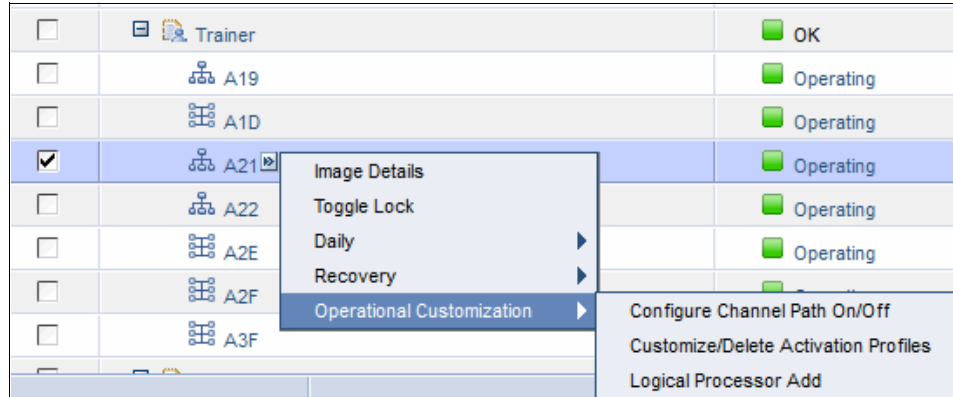


Figure 4-2 Logical Processor Add function

This opens the window shown in Figure 4-3. Make the adjustments that you want and then select one of these options:

**Change running system** If you want to change the number of CPs available to the current running system, but do *not* want the change to be saved in the activation profile for that LPAR.

**Save and change** If you want to change the running system and save the changes to the activation profile so that the changes *will* be retained across a future deactivation and reactivation.

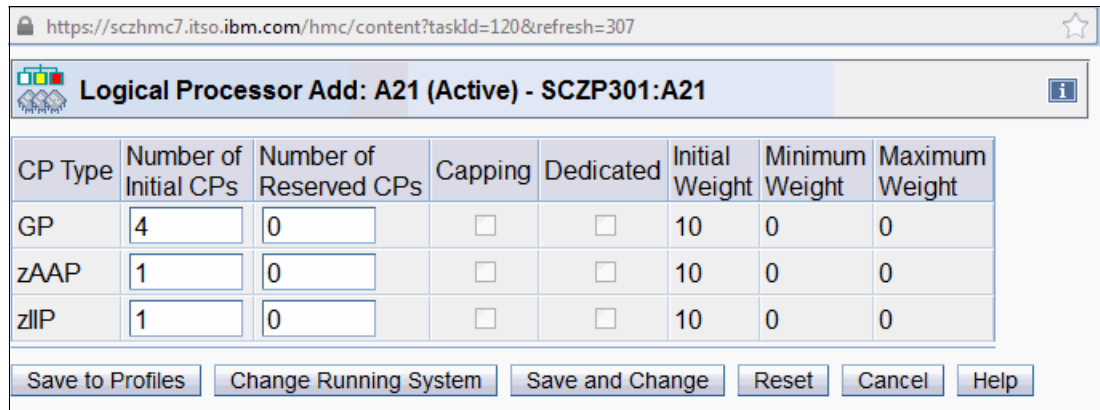


Figure 4-3 Updating INITIAL and RESERVED values dynamically

When you select **Change Running System** or **Save and Change**, you see message ISN011I on the system console for that system, indicating that the additional CPs have been added.

Be aware of the PRESCPU parameter in IEASYSxx. If that keyword *is* specified, when you IPL, the system will come back up with the same number of CPs that it had before the IPL. If that parameter is not specified, the number of CPs that are online after the IPL is determined by the INITIAL value in the LPAR activation profile.

#### 4.11.5 Rollback

This function is not rolled back to prior releases of z/OS, or to CPCs before z10.

#### 4.11.6 Further information

For more information about using dynamic CPU addition, see *z/OS MVS Planning: Operations*, SA23-1390.

For more information about the DYNCPADD parameter in the LOADxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

### 4.12 Switching between primary and secondary DASD without an IPL

You no longer need an IPL when switching between primary and secondary DASD.

#### 4.12.1 The issue

With today's larger DASD control units and the use of SMS to allocate data sets across many devices and control units, the failure of a DASD control unit can affect the whole sysplex and potential even multiple sysplexes. Peer-to-Peer Remote Copy (PPRC) lets you maintain a duplex relationship between primary and secondary DASD volumes. The use of PPRC means that you can use the secondary DASD if a failure takes out the primary control unit or volume. However, the failure would more than likely cripple all connected systems, and even if it did not, replacing the primary volumes with the secondaries would be disruptive.

HyperSwap is an ability to dynamically and non-disruptively switch from the primary to secondary volumes. It can do so either in a planned manner (in preparation for a configuration change, for example), or in response to a failure affecting one or more primary volumes.

IBM provides two types of HyperSwap. One is provided as part of the Geographically Dispersed Parallel Sysplex (GDPS) offering. The other is in a z/OS component called Basic HyperSwap. Basic HyperSwap was delivered as a SPE in z/OS 1.9. It consists of two IOS address spaces. One contains the functional HyperSwap code of TotalStorage Productivity Center for Replication (TPC-R) and a second manages communications between all configured z/OS images within a Base or Parallel Sysplex. For Basic HyperSwap, planned outages are handled with TPC-R, and unplanned outages are handled by IOS only.

#### 4.12.2 How this is addressed in z/OS 1.10

The HyperSwap function enables I/O to the primary logical devices in a synchronous Peer-to-Peer Remote Copy consistency group to be swapped, *without human intervention*, to the secondary logical devices in the Peer-to-Peer Remote Copy consistency group. This process causes minimal disruption to the applications that are using the logical devices.

The **SETHS** command and the **Display HS** became available with z/OS 1.10.

- ▶ **SETHS** command is used to manage the HyperSwap function.
- ▶ **Display HS** is used to show information about the HyperSwap function and the status of device pairs in the PPRC configuration.

### 4.12.3 Restrictions or considerations

The MIH IOTIMING in IECIOSxx member can be used to trigger an unplanned HyperSwap. Review the options that fit your installation. To display current options, issue the following command:

```
D IOS,MIH
```

### 4.12.4 Rollback

APAR OA20658/OA25299 implemented Basic HyperSwap.

The **SETHS** and **D HS** command are rolled back to z/OS 1.9.

### 4.12.5 Further information

For more information about the use of the **SETHS** and **D HS** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the IECIOSxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about Basic HyperSwap, see *IBM DS8000 and z/OS Basic HyperSwap*, REDP-4441, available on the web at:

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4441.pdf>

## 4.13 Specify NOBUFFS action (SMF) at the log stream level

This section describes enhancements to the specify NOBUFFS action at the log stream level.

### 4.13.1 The issue

SMF to log streams has been around a while. Unfortunately, there have not been as many controls available in SMFPRMxx as there were with the old VSAM formats. When using VSAM, you would receive warnings as SMF began buffering records, generally at some percentage that you chose, because SMF could not update the MANx data sets. You then had enough time to react to the cause of the failure.

### 4.13.2 How this is addressed in z/OS 1.12

The addition of the NOBUFFS parameter on the LOGSTREAM DEFAULTLSNAME or the LSNAME statement in SMPPRMxx allows you to specify what behavior you want SMF to take when records begin buffering. BUFUSEWARN allows you to specify a percent full and NOBUFFS specifies whether you want to write a message about it or to enter a restartable

wait state. BUFSIZMAX can be specified to increase the default buffer size from the default of 128 M to a maximum of 1 G.

When the BUFUSEWARN percentage is reached, SMF generates IFA785E telling you how much of the current buffer space is used for that log stream.

With NOBUFFS(MSG), SMF generates IFA786W to warn you the buffer for a particular log stream has been exhausted.

With NOBUFFS(WAIT), SMF forces the system into a restartable wait state.

After you update the SMFPRMxx entry, you can issue the **SET SMF=xx** or the **SETSMF** command to change the entries dynamically.

### 4.13.3 Restrictions or considerations

IF BUFMAXSIZE is specified and the amount of storage is not available at SMF start, SMF decreases the size by 20% and tries again. It does this up to four more times. Message IFA723E is generated indicating the amount of storage SMF was finally able to obtain.

### 4.13.4 Rollback

These options have not been rolled back to previous releases.

### 4.13.5 Further information

For more information, see *z/OS MVS System Management Facilities (SMF)*, SA38-0667.

For details about coding SMFPRMxx, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 4.14 Forcing hung commands to avoid an IPL

A **FORCE** parameter has been added to the **CMDS** command to allow you to avoid performing an IPL to clear up hung non-abendable commands.

### 4.14.1 The issue

Before z/OS 1.13, you could delete a command that had not started yet, or cancel (abend) a running command if it did not have the non-abendable attribute. However, removing a non-abendable task required an IPL.

### 4.14.2 How this is addressed in z/OS 1.13

A new parameter, **FORCE**, is added to the **CMDS** command. When you issue a **CMDS FORCE** command, the non-abendable attribute is overridden and the command can stop as usual. By separating the **ABEND** and **FORCE** options, you can define different Resource Access Control Facility (IBM RACF®) profiles to allow an operator to issue a **CMDS ABEND** command, but not a **CMDS FORCE** command. Use **FORCE** when the only alternative is to IPL the system again.

Both the ABEND and FORCE options requires subparameters CMD= and ID=. The system then stops the command that CMD=cccccc and ID=nnnn identifies and issues one of the following messages:

ABEND cause ABEND code 422, reason code 00010301

FORCE cause ABEND code 422, reason code 00010302

The system issues message IEE064I in response to this command. It does not send any messages to the console that issued the **CMDS** command. Therefore, you need to review the IEE064I message in SYSLOG.

Use the ABEND option with caution, being careful to avoid leaving the system in an inconsistent state. For example, if the command you intend to cancel is the first of three commands to complete a task, you need to understand the impact if the second and third commands are run, but the first one is not.

**Note:** The ABEND request will be rejected with message CNZ6002I if the command is in a non-abendable state.

Use the FORCE option with extreme caution, bearing in mind that you might be stopping a command that is updating critical system data. Use this parameter only as a last resort, such as when an IPL is needed if the command is not stopped.

### 4.14.3 Restrictions or considerations

If you want to take a memory dump to help debug a perceived problem, issue a **CMDS DUMP** command before you issue the **CMDS ABEND** or **CMDS FORCE** commands. Note that **CMDS DUMP** will dump only the MASTER and CONSOLE address spaces. If you believe the target command might have functions running in another address space, use the **DUMP** command instead, specifying that address space, along with MASTER and CONSOLE.

### 4.14.4 Rollback

This function is not available before z/OS 1.13.

### 4.14.5 Further information

For more information about the use of the **CMDS** command, see *z/OS MVS System Commands*, SA38-0666.

## 4.15 Dynamically configure storage-class memory (SCM)

This section details enhancements that allow you to dynamically configure storage-class memory (SCM).

### 4.15.1 The issue

As real storage is getting larger and many products are using 2 GB data spaces and requiring 1 M paging due to their use of 64-bit storage, auxiliary storage is becoming the bottleneck.



## 4.15.2 How this is addressed in z/OS 1.13

The **PAGESCM** parameter in IEASYSxx member specifies the minimum amount of SCM (also known as flash memory) that should be made available for use as auxiliary storage. The system reserves this amount of SCM during IPL for subsequent use as auxiliary storage. More SCM is allocated on an as-needed basis if use of this initial amount of SCM is exceeded. The default value is ALL, meaning that all SCM is reserved for paging at IPL.

The **CONFIG SCM** command is used to reconfigure SCM, both logically and physically. To bring SCM online, an amount must be specified. To take SCM offline, a range of starting and ending addresses of the SCM blocks must be specified. The commands to do this are:

```
SCM(ddddddddM|G|T),ONLINE|ON  
SCM(ddddddddM|G|T),OFFLINE|OFF  
SCMscm_ranges,OFFLINE|OFF
```

To obtain information about how SCM is configured, use the **D M=STOR** command.

## 4.15.3 Restrictions or considerations

Requires EC12 hardware family or later to support flash memory.

## 4.15.4 Rollback

This function is available on z/OS 1.13 with the web deliverable z/OS 1.13 RSM Enablement Offering. It is incorporated into z/OS 2.1.

## 4.15.5 Further information

For more information about the use of the **CONFIG SCM** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the **PAGESCM** parameter in the IEASYSxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

# 4.16 End a task with new operand on FORCE

A new operand has been added to the **FORCE** command.

## 4.16.1 The issue

Before z/OS 2.1, it was not possible to cancel a subtask of a job. Therefore, the job had to be canceled. If the cancel does not work, the job needs to be FORCE'd, which can in some cases cause a requirement to IPL.

## 4.16.2 How this is addressed in z/OS 2.1

In z/OS 2.1, a new operand is available for the **FORCE** command to enable you to specify the TCB address of a particular task for the system to stop. This function is intended to be used to preserve system availability when a task holds resources that are required by other critical

functions when there seems to be no other alternative to IPL. The new format of the force command is **FORCE jobname,TCB=tttttt**.

To control the usage of this new option, update the security profile with the following:

MVS.FORCETCB.\* SAF profiles in OPERCMDS class

**Note:** The **FORCE jobname,TCB=tttttt** command is a functional replacement for a program called CALLRTM that was traditionally provided by IBM Support. CALLRTM allows a system programmer to end a specific TCB in the system. However, the new **FORCE jobname,TCB=tttttt** command is an easier way to achieve the same result. Additionally, the new **FORCE** parameter causes an ABEND80D to be issued against the target TCB. The abend is retryable with the new **RETRY=NO|YES** option on **FORCE TCB**.

### 4.16.3 Restrictions or considerations

It is important that you read and understand the restrictions that are related to the **FORCE** command before using it.

**Important:** Never use the **FORCE** command without understanding that its use could result in having to IPL that system again.

### 4.16.4 Rollback

This function is not available for releases before z/OS 2.1.

### 4.16.5 Further information

For more information about the use of the **FORCE** command, see *z/OS MVS System Commands*, SA38-0666.

## 4.17 Auto-reply for WTOR

An new function called Auto Reply has been added for WTOR.

### 4.17.1 The issue

z/OS clients typically use a system automation product to reply to known “write to operator with replies” (WTORs). However, these automation products are not available during the early stages in the IPL process, and so are unable to respond to any WTORs produced later.

An example of such a message might be an IEE357A message that indicates an error in the SMF parameters. The only way to automatically respond to such messages was to create an MPF exit.

### 4.17.2 How this is addressed in z/OS 1.12

z/OS 1.12 provided a new function called Auto Reply. The Auto Reply function uses a policy defined in a PARMLIB member to identify known WTORs, and automatically reply to those

WTORs if the operator or your system automation product does not reply within a defined amount of time.

IBM provides a default policy that you can add to or adjust. The default policy is activated during IPL unless explicitly requested by the user to not activate. This allows WTORs issued during NIP to be automated. Operator commands are provided that allow the activation and deactivation of the auto-reply policy, display the auto-reply policy and the list of current WTORs that are being monitored by auto-reply processing, and cause auto-reply processing to stop the monitoring of a specific outstanding WTOR. A new system parameter is provided that can be specified in the IEASYSxx PARMLIB member or in response to message IEA101A. This parameter allows an installation to provide the set of PARMLIB members that contain an auto-reply policy or to request that auto-reply processing should not be activated.

### 4.17.3 Restrictions or considerations

If you do not want to use this new capability, you *must* specify AUTOR=OFF in IEASYSxx. Do not remove the AUTOR00 member because this might cause problems if future maintenance activities attempt to change it. Also, create a new AUTORxx for your customization and concatenate it in the IEASYSxx member:

```
AUTOR=(xx,00)
```

Use the **Display AUTOR** command to view the active policy detail or WTOR status.

### 4.17.4 Rollback

This function is not available before z/OS 1.12.

### 4.17.5 Further information

For more information about the use of the **D AUTOR** and **T AUTOR** commands, see *z/OS MVS System Commands*, SA38-0666

For more information about the use of the AUTORxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.18 DCCF support for WTOR Auto-Reply

DCCF support has been added for WTOR Auto-Reply.

### 4.18.1 The issue

Before z/OS 2.1, the Auto-reply function could not respond to a synchronous WTOR.

These WTOR messages went to only one console that could reply. If SYNCHDEST is specified in the CONSOLxx member, the WTOR could be sent to specific active MCS consoles. If it is not specified, the message only went to the System Console (HMC).

Synchronous WTORs indicate critical problems. The system is stopped until a reply is provided, and any delay in replying is likely to affect other systems in the sysplex.

A synchronous WTOR is displayed with these restrictions:

- ▶ Only on consoles attached to the system that issues the WTOR
- ▶ Only on one operator console at a time (more on this later)
- ▶ The WTOR is moved to another console (on the same system) if a reply is not entered within approximately two minutes
- ▶ A reply is only accepted from the console on which the WTOR is *currently* displayed
- ▶ Operator consoles not involved with the WTOR process appear to be hung

## 4.18.2 How this is addressed in z/OS 2.1

There are two enhancements available for synchronous WTOR processing using the Disabled Consoles Communication Facility (DCCF) synchronous WTOR:

- ▶ The first is designed to extend the Auto Reply function that was introduced in z/OS V1.12 to allow it to respond to WTORs displayed through synchronous WTOR.
- ▶ The second is intended to notify all locally attached MCS consoles about the current destination of a WTOR displayed by synchronous WTOR. This makes it easier and faster to locate the console on which the response might be entered.

For synchronous WTORs, review the sample AUTOR00 that is available in SYS1.IBM.PARMLIB.

Use the **Display AUTOR** command to view the active policy detail or WTOR status.

Use the **Display Console** command to view the current SYNCDEST setting.

## 4.18.3 Restrictions or considerations

For synchronous WTORs, the reply in the AUTORxx member should not contain any system symbols because they will not be resolved when the reply is issued. It is suggested that the delay value be less than the XCF's failure detection interval. This is to avoid SFM partitioning the system out of the sysplex because the system is waiting for a reply to the WTOR.

Use the SYNCHDEST option in CONSOLxx member to allow the reply to a synchronous WTOR on a z/OS console per SYNCHDEST versus the System Console (HMC) only.

## 4.18.4 Rollback

This function is not available before z/OS 2.1.

## 4.18.5 Further information

For more information about the use of the AUTORxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 4.19 New MODIFY VLF command

The MODIFY VLF command has been added to allow you to replace your configuration.

### 4.19.1 The issue

Virtual lookaside facility (VLF) is an MVS service that enables applications to minimize I/O operations for frequently retrieved objects. The member COFVLF00 specifies the VLF Class and Major Name that are used for objects. To make a change, you had to bounce VLF, which lost all data used to minimize I/O.

### 4.19.2 How this is addressed in z/OS 2.1

The new **Modify VLF,Replace** command allows VLF to replace the current configuration with a new configuration. The 'xx' value can be a single COFVLFxx member of the logical PARMLIB, or a concatenated list of up to 16 COFVLFxx PARMLIB members, enclosed within parentheses and separated by commas. For example: **MODIFY VLF,REPLACE,NN=(AA,BB,CC)**.

To display the VLF Class and MaxVirt, use Health Checker with Verbose option of VLF\_MAXVIRT check. No command is available for the individual entries per VLF class.

### 4.19.3 Restrictions or considerations

The Performance group should review VLF statistics found in SMF record 41, subtype 3, field SMF41TRIM, to determine whether any trimming is occurring.

If **Modify VLF** is used to implement dynamic options, you need to update the COFVLFxx member to avoid regressing options at IPL time. VLF\_MAXVIRT became available in z/OS 2.1.

#### Monitor usage of VLF with Health Checker

Health Checker checks to see whether the VLF is trimming recently added objects to make room for new objects. If so, the MAXVIRT setting for at least one VLF class might be too small for VLF to provide a good performance benefit.

VLF\_MAXVIRT

### 4.19.4 Rollback

This function is not available before z/OS 2.1.

### 4.19.5 Further information

For more information about the use of the **Modify VLF** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the COFVLFxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.20 Add/Remove MCS consoles dynamically

Enhancements now allow you to add and remove MCS consoles dynamically.

### 4.20.1 The issue

In a sysplex environment, the limit of consoles you can define varies depending on the operation mode of the console support. In shared mode, up to 99 MCS, SMCS, and subsystem consoles can be defined per sysplex. In distributed mode, up to 250 MCS, SMCS, and subsystem consoles can be defined per z/OS image.

Before z/OS 2.1, to remove obsolete consoles, you had to do a Sysplex IPL. Adding a console required updating the CONSOLxx member and scheduling an IPL on the images required. Therefore, a rolling IPL worked.

### 4.20.2 How this is addressed in z/OS 2.1

In z/OS 2.1, support enables you to add and remove MCS consoles dynamically when they are being used in distributed mode. **SETCON** command processing is designed to process a CONSOLxx PARMLIB member and add new consoles, up to the system and sysplex limits for the maximum number of consoles. The **SETCON** command is designed to let you specify a console to be removed.

Example 4-6 shows the process.

*Example 4-6 Dynamically deleting and defining consoles*

---

```
D CONSOLES,NACTIVE,CN=#@$AF480
CNZ4100I 10.47.15 CONSOLE DISPLAY 091
CONSOLES MATCHING COMMAND: D CONSOLES,NACTIVE,CN=#@$AF480
MSG:CURR=0    LIM=3000 RPLY:CURR=1    LIM=999    SYS=#@$A    PFK=00
NAME      TYPE    STATUS      DEFINED      MATCHED
#@$AF480  MCS      INACT      #@$A        #@$A

SETCON DELETE,CN=#@$AF480
CNZ4300I MCS CONSOLE #@$AF480 HAS BEEN REMOVED

D CONSOLES,NACTIVE,CN=#@$AF480
IEE274I DISPLAY CONSOLE #@$AF480 NOT VALID

SET CON=00
CNZ6003I COMMAND ACCEPTED FOR EXECUTION: SET CON=00
IEE252I MEMBER CONSOLOO FOUND IN SYS1.PARMLIB
CNZ6004I SYSCONS CANNOT BE ADDED DYNAMICALLY
IEA196I CONSOLOO F400: DEVNUM ALREADY DEFINED. STATEMENT IGNORED.

D CONSOLES,NACTIVE,CN=#@$AF480
CNZ4100I 10.47.45 CONSOLE DISPLAY 112
CONSOLES MATCHING COMMAND: D CONSOLES,NACTIVE,CN=#@$AF480
MSG:CURR=0    LIM=3000 RPLY:CURR=1    LIM=999    SYS=#@$A    PFK=00
NAME      TYPE    STATUS      DEFINED      MATCHED
#@$AF480  MCS      INACT      #@$A        #@$A
```

---

### 4.20.3 How this is addressed in z/OS 1.10

A console service allows you to remove an MCS, SMCS, or subsystem console from a system or sysplex. The service is available as a sample job in SAMPLIB member IEARELCN.

This display command shows all consoles in use:

D C,L

### 4.20.4 Restrictions or considerations

Having consoles in Distributed mode is not required by z/OS, but is required for new function to add/remove a console in z/OS 1.12.

#### **Monitor usage of Consoles with Health Checker**

Health Checker ensures that there are not an excessive number of inactive EMCS consoles. **CNZ\_EMCS\_Inactive\_Consoles** became available in z/OS 1.4.

The check identifies installations that have not explicitly identified their console service operating mode. **ZOSMIGV1R13\_CNZ\_Cons\_Oper\_Mode** became available in z/OS 1.13.

### 4.20.5 Rollback

This full function is not available before z/OS 2.1, but a limited console service became available in z/OS 1.10.

### 4.20.6 Further information

For more information about the use of the **SETCON** commands, see *z/OS MVS System Commands*, SA38-0666.

For information about the Console Service, see *z/OS MVS Planning: Operations*, SA23-1390.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.21 New Message Flood Automation

z/OS 1.9 introduced the new Message Flood Automation feature.

### 4.21.1 The issue

A job sends many messages (called message flooding) to the console due to a program bug, error condition, or a large quantity of items that then must be processed by the subsystems. GDPS built an MPF exit process to handle the valid number of DASD devices in a short window because this was more common in its scripts.

### 4.21.2 How this is addressed in z/OS 1.9

With z/OS 1.9, Message Flood Automation is used to control messages flooding the console by the installation message policy member MSGFLDxx. The policy eliminates large number

of console messages that reduces CPU cycles in MPF processes. Originally, it was implemented as a IEAVMXIT exit and command exits.

Message Flood Automation requires two parts: Build MSGFLDxx member, then a **Start MFA** command that references that member.

There are multiple commands to support the MFA process: **SETMF**, **SET MSGFLD**, and **DISPLAY MSGFLD**

Start it: **SET MSGFLD=xx** then **SETMF ON**

Stop it: **SETMF OFF** or part: **SETMF MONITOROFF**

### 4.21.3 How this is addressed in z/OS 1.11

With z/OS 1.11 base, the MFA became part of the internal message processing and eliminated the use of IEAVMXIT and command exit. These changes make it easier to install Message Flood Automation and allowing Message Flood Automation to take actions against all messages. You no longer can exclude Message Flood Automation from seeing MPF messages.

### 4.21.4 How this is addressed in z/OS 1.13

With z/OS 1.13, you specify the initial Message Flood Automation MSGFLDxx PARMLIB member through the MSGFLD(xx,[(wrd)]) parameter on the CONSOLxx INIT statement. This replaces the current method of issuing a series of **SET MSGFLDxx** and **SETMF ON** commands. You need the PTF for APAR OA33652 to allow the MSGFLD parameter on the CONSOLxx INIT statement to be tolerated (ignored) on z/OS 1.12 and z/OS 1.11 systems when sharing a CONSOLxx PARMLIB member with z/OS 1.13.

Example 4-7 shows enabling MFA at IPL.

*Example 4-7 Enable MFA at IPL time without issuing commands*

---

To load and enable MSGFLD00, you only need to specify the MSGFLD(00,(ON)) parameter explicitly.

---

### 4.21.5 Restrictions or considerations

The **Display MSGFLD** gives the status on MFA functions: MF status and Message Rate monitoring status.

#### **Monitor usage of MFA implementation with Health Checker**

Health Checker checks to see whether an obsolete version of Message Flood Automation is installed. z/OS 1.11 eliminates the use of message exit IEAVMXIT and the command exit CNZZCMXT. If either message exit IEAVMXIT or command exit CNZZCMXT are installed on a z/OS V1 R11 system, the check generates an exception.

**CNZ\_OBSOLETE\_MSGFLD\_AUTOMATION** became available in z/OS 1.11.



## 4.21.6 Rollback

MFA installed with a SPE rolled back to z/OS 1.6 as a IEAVMXIT exit.

MFA change to internal message processing rolled back to z/OS 1.9 with APAR OA25602.

## 4.21.7 Further information

For more information about the use of the **SET MSGFLD=xx** and **SETMF** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the MSGFLDxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.22 Added LPA defer wait capability

This section covers the new LPA defer wait capability.

### 4.22.1 The issue

More products are using PDSE and want to add them to the LPA for single copy and eliminate I/O to DASD. Before z/OS 1.12, you could add PDSE program objects to LPA at the tail end of the IPL (COMMNDxx PARMLIB member). However, an application had no way of knowing whether things were ready for it. This left it up to you to sequence these events.

### 4.22.2 How this is addressed in z/OS 1.12

With z/OS 1.12, you can have LPA ADD statements in your IPL-time PROGxx specification (before z/OS 1.12, these specifications were not processed). Now they will be, at a deferred point late in the IPL. SMSPDSE must be initialized.

Use the following command to display this:

```
D PROG,LPA
```

### 4.22.3 Restrictions or considerations

The new function is provided for an application to ask that it wait until all of the deferred ADD processing is complete. This removes your requirement to sequence the events.

### 4.22.4 Rollback

This function is not available before z/OS 1.12.

## 4.22.5 Further information

For more information about the use of the PROGxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the **D PROG,LPA** commands, see *z/OS MVS System Commands*, SA38-0666.

## 4.23 Auto-IPL using DIAGxx

You can now use DIAGxx to automatically IPL a system.

### 4.23.1 The issue

Many installations use the HMC to initiate an IPL of a system. These are typically in locked rooms that are often hundreds or even thousands of miles away, which requires extra security and network connectivity. There are times when either the security or the network can fail. Using AutoIPL can expedite the process and provide consistent and reproducible results. The action is taken automatically if there is a system disabled wait based on the wait state action table (WSAT) in the nucleus or if initiated by a system command.

### 4.23.2 How this is addressed in z/OS 1.10

Update DIAGxx (referenced in IEASYSxx or by command) that specifies the location of your stand-alone memory dump and your z/OS sysres allows you to control how z/OS responds to an operation action or a system wait. This includes the IPL volume or LAST (for last used) and the LOAD parameters (IODEF, IEASYSxx, Prompt level, Alternate Nucleus), and can also include stand-alone memory dump information.

After the DIAG member is set (by using the **SET DIAG=xx** command), you can then run a normal system shutdown. To initiate the Auto-IPL, issue this command:

```
VARY XCF,sysname,OFFLINE,[SADUMP,]REIPL
```

If entered with both SADUMP and REIPL, a stand-alone memory dump is started and upon completion, the z/OS system goes through IPL again:

```
AUTOIPL SADMP(sadmp info) MVS(mvs info) where:  
SADMP information includes the device and loadparm or NONE  
MVS information includes the device|last and loadparm or NONE
```

### 4.23.3 Restrictions or considerations

If z/OS is about to enter a disabled wait, the action that is specified in the DIAG member **ACTIVE AT THE TIME** is followed. To determine what DIAGxx is active, issue the **DISPLAY DIAG** command. To deactivate AutoIPL, code **AUTOIPL SADMP(NONE) MVS(NONE)** in DIAGxx and issue the **SET DIAG=xx** command.

**Consideration:** AutoIPL is not appropriate for a GDPS environment.

The required hardware is necessary to support the function. That includes z10EC or z9EC with feature code 9904 *and* driver 67 or later. Message IGV010I appears if some or all of the required support is not present.

If Sysplex Failure Management is active, AutoIPL might be delayed while SFM runs the necessary fencing. The failed system might appear to be hung.

### Monitor usage of AUTOIPL with Health Checker

Health Checker checks whether the customer environment can support an AutoIPL policy and if it is, determines whether the AutoIPL policy is active. **SVA\_AUTOIPL\_DEFINED** became available in z/OS 1.11.

Health Checker also runs device validation for devices specified in the AutoIPL policy for SADMP or MVS when an AutoIPL policy exists. It reports problems if the device validation fails. If the check determines that there is no AutoIPL policy defined, or that the customer does not have the appropriate hardware feature, the check stops running.

SVA\_AUTOIPL\_DEV\_VALIDATION

## 4.23.4 Rollback

This function is not available before z/OS 1.10.

## 4.23.5 Further information

For more information about the use of the **Vary** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the DIAGxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the AutoIPL and list of wait states, see *z/OS MVS Planning: Operations*, SA23-1390.

For more information about the IBM Health Checker, see *IBM Health Checker for z/OS Users Guide*, SC23-6843.

## 4.24 z/OS reusable address space support and usage

Enhancements have been made in the area of z/OS reusable address space support and usage.

### 4.24.1 The issue

Running out of address space IDs (ASIDs) is a cause of IPLs. If you can reduce the rate at which you consume ASIDs, you can lengthen the time between IPLs. You indicate how many ASIDs are allowed on your system using the following parameters:

- ▶ IEASYSxx's RSVNONR parameter

This indicates the number of ASIDs that are reserved in the address space vector table (ASVT) for replacing ASIDs that become non-reusable. The more times that you plan to restart programs that make their address space non-reusable during the life of the IPL, the higher this number should be. The default RSVNONR value is 100.

Address spaces get marked as being non-reusable when a program that has been using cross memory services ends. When such a program ends, the system ends the address space and marks its associated ASVT entry as being non-reusable (unavailable) until all of the address spaces the program had cross memory binds with have ended. At that point, the ASID is made available for use again. This action protects data integrity by ensuring that a new program cannot be started in that address space and start to use the cross memory links that were in place from the previous program.

- ▶ IEASYSxx's MAXUSER parameter

This indicates the maximum number of concurrent jobs and started tasks. The default is 255.

- ▶ IEASYSxx's RSVSTRT parameter

This is the maximum number of address spaces to be reserved in the ASVT for START commands that are issued after the MAXUSER number of concurrent address spaces has been reached. The default is 5.

The maximum total number of ASIDs on a system is the sum of RSVNONR, MAXUSER, and RSVSTRT, and cannot be larger than 32767. If you have many programs that cause their address space to be marked as non-reusable, specify a larger RSVNONR number. However, doing so reduces the number of available ASIDs. Also, setting these values *too* high can result in a wait state during IPL due to an excessive amount of storage being used.

## 4.24.2 How this is addressed since z/OS 1.9

z/OS 1.9, combined with changes in z990 and later CPCs, allows you to use cross memory services without having a data integrity exposure if another program uses that address space. Basically, it is now possible to indicate that address spaces that will be using cross memory services can be assigned a unique instance number when it is started. The instance number allows connected address spaces to determine whether the program they are communicating with is the one that they expect to be there.

In support of this change, starting with z/OS 1.9, there are two kinds of ASIDs: Ordinary and reusable. All ASIDs start out as ordinary, but some might later be marked as being reusable. If there are no available ASIDs marked as reusable, and one is requested, the system converts an ordinary ASID to a reusable ASID. After an ASID is marked as reusable, it will remain as reusable. When the program in that address space ends, the ASID goes back into a pool of available reusable ASIDs.

An address space that is created with the **START** command specifying REUSASID=YES, or started using the ASCRE macro with ATTR=REUSASID is designated as a reusable address space.

To use a reusable address space, the programs in that address space and any programs they communicate with must conform to certain programming rules that are documented in *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614. If a program that does not conform to those rules is started in a reusable address space, it is likely to end with an ABEND0D3. Therefore, only specify REUSASID=YES on the **START** command for programs or products that have explicitly stated that they support the use of reusable address spaces.

You can control system-wide whether you can use reusable ASIDs by using the PARMLIB member DIAGxx. The REUSASID parameter was added to DIAGxx in z/OS 1.9 and the default was NO. As of z/OS 1.12, the default was changed to YES. However, if you want to use this support, REUSASID=YES must still be specified on the START command. This is because many programs still do not support this capability and would abend if started in a reusable address space.

## A processing example

In an extreme example to illustrate this new function, imagine the following scenario:

1. You specify RSVNONR as 12, MAXUSER of 50, and RSVSTRT of 0. Therefore, you have a maximum of 62 ASIDs on the system.
2. You start 40 address spaces that will use cross memory services. When or if those address spaces end, 40 ASIDs are marked as being non-reusable. You have 10 jobs or started tasks available before you reach the MAXUSER limit of 50. Of the 62 ASIDs that you started out with:
  - 40 are marked non-reusable when they end,
  - 12 are reserved to replace non-reusable ASIDs (as indicated by the RSVNONR of 12 value you specified)
  - 10 are ordinary, and can be marked either reusable or non-reusable depending on what occurs on the system. They might remain ordinary if they do not meet the criteria for being marked as non-reusable or are not started as reusable.
3. You now start 10 address spaces with the REUSASID=YES keyword on the **START** command. The 10 ASIDs for those tasks are converted from ordinary to reusable in the ASVT. You have reached the MAXUSER value on the system (40 non-reusable + 10 reusable = 50). The 62 ASIDs are then in these states:
  - 40 are marked non-reusable when they end,
  - 12 are reserved to replace non-reusable ASIDs (as indicated by the RSVNONR value you specified),
  - 10 are marked reusable.All available ASIDs are currently in use in the ASVT.
4. Imagine that five of the reusable started tasks are stopped. Those five reusable ASIDs are now only available to other *reusable* started tasks that might need them. If the next started task you attempt to start needs an ordinary ASID, none are available and the started task does not start. However, if the next started task is started as being reusable, it can use one of the available five reusable ASIDs.

This example illustrates the importance of careful planning for the RSVNONR, RSVSTRT, and MAXUSER values that you specify. You must take into account which of your programs and product can use reusable address spaces.

Over time, more programs have been modified to be able to use reusable address spaces. This allows you to lengthen the lifetime of an IPL if these address spaces are stopped and restarted. Without this support, these address spaces otherwise mark an address space to be non-reusable ASID. Eventually you will run out of ASIDs, forcing an IPL. The programs that support reusable address spaces at the time of writing of this book are:

- ▶ z/OS 1.9 added support for reusable address spaces to VLF, LLA, DLF, and CATALOG.
- ▶ z/OS 1.10 added support for reusable address spaces to Communications Server RESOLVER, TCP/IP address spaces, DFSMSrmm, TN3270, and HWIBCPii (BCPii).
- ▶ z/OS 1.13 added support for reusable address spaces to DEVMAN (device support address space in DFSMS).

### 4.24.3 Restrictions or considerations

Because an ordinary ASID that has been marked reusable remains reusable, reducing the number of ordinary ASIDs available for other started tasks and jobs, do not specify REUSASID=YES, or ATTR=REUSAID unless you need to. You want to ensure that you have

some ordinary ASIDs left for non-reusable tasks. Keep this in mind when setting your RSVNONR value.

In z/OS 1.9, an IBM Health Checker for z/OS health check (IEA\_ASIDS) was added that helps identify how many ordinary and reusable ASIDs are being used. One useful item in this check is that it can tell you how many non-reusable ASIDs you have used each day (on average) since the system IPL. In addition, the check projects approximately how many days you have until you need to IPL again. Figure 4-4 contains a sample of this information.

```
IEAVEH061I The system has been IPLed for between 4 and 5 days. On the
average 7 ASIDs have become non-reusable per day. At the current rate of
depletion, the system will run out of ASIDs in 606 days.
```

Figure 4-4 IEA\_ASIDS sample output for rate of depletion of non-reusable ASIDs

**Tip:** Specify the VERBOSE option for health check IEA\_ASIDS to see more information about individual connections to non-reusable ASIDs.

At the time of writing, some address spaces do not use the REUSASID support. Therefore, on a system where you stop and restart these address, more ASIDs are needed if you want to lengthen your time between IPLs. Most of these address spaces do not require frequent stopping and restarting:

- ▶ DB2: The four address spaces associated with DB2: xxxxMSTR, xxxxDBM1, xxxxIRLM, and xxxxDIST.
- ▶ IBM Health Checker for z/OS: The default proc name is HZSPROC.
- ▶ Generic Tracker: GTZ.
- ▶ System REXX: AXR, and AXR00-08.

#### 4.24.4 Rollback

There is no rollback to earlier releases of either the reusable address space support or for the programs that use that capability.

#### 4.24.5 Further information

For more information about the changes in the area of non-reusable address spaces, see the following documentation:

- ▶ The section titled “Reusing ASIDs” in *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614.
- ▶ *z/OS MVS Initialization and Tuning Reference*, SA23-1380.
- ▶ The section titled “Non-reusable address spaces” in *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

## 4.25 z/OS report on linkage indexes

Linkage indexes, and their effect on availability, are described in the *z/OS Planned Outage Avoidance Checklist*, SG24-7328. This section briefly describes them, along with the related IBM Health Checker for z/OS check that helps report on their usage.

In a cross memory environment, tables and linkages connect the service provider's address space to the user's address space and to the tables and linkages that provide the necessary authorization for the service provider. The linkage index (LX) locates a specific entry in the linkage table. Non-system LXs connect an entry table to the linkage table in one or more, but not all, address spaces. System LXs connect an entry table to all linkage tables.

Running out of linkage indexes is a cause for an IPL. Increasing the total number of linkage indexes that are allowed, and having system linkage indexes marked as being reusable, help to lengthen the life of an IPL.

The limit on the total number of linkage indexes (LXs) has not changed since z/OS 1.7:

- ▶ The total number of LXs can be up to 2048 on a system that is running on a processor before either a z990 or z890 processor at driver level 55. (Note that z/OS 2.1 requires at least a IBM z9® EC or z9 BC processor.)
- ▶ The total number of LXs can be up to 32768 if the system is running on a z890 or z990 processor at driver level 55 or later. LXs exceeding 2048 are known as extended LXs.

As of z/OS 1.6, LXs are able to be reused. This allows you to reuse both system linkage indexes and non-system linkage indexes. This means that the original requester of the LX can choose to reconnect to the LX if the address space stops and then restarts. Using this technique allows you to run longer between IPLs

### 4.25.1 The issue

Starting with z/OS 1.3, the system issues messages that indicate if the system is depleting linkage indexes. These messages are IEA063E, IEA065E, IEA066I, IEA070E, IEA071I, IEA072E, or IEA073I, and should be included in your system automation product. However, there was no easy way to determine the number of linkage indexes that are in use or the total number that are available.

### 4.25.2 How this is addressed in z/OS 1.9

Starting with z/OS 1.9, IBM Health Checker for z/OS health check **IEA\_LXS** provides a report on the use of system and non-system LXs, both normal and extended. As with the health check **IEA\_ASIDS**, the limit, the number that are available, the number that are in use, and total number are provided.

### 4.25.3 Restrictions or considerations

Turn VERBOSE on for health check **IEA\_LXS** to get more information about each LX: The LX value, the ASID, and job name.

### 4.25.4 Rollback

There is no rollback provided for this IBM Health Checker for z/OS check.

### 4.25.5 Further information

For more information about linkage indexes, see these resources:

- ▶ *z/OS Planned Outage Avoidance Checklist*, SG24-7328
- ▶ *z/OS MVS Initialization and Tuning Reference*, SA23-1380
- ▶ *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614





## **z/OS enhancements: GRS**

This chapter provides information about enhancements in z/OS Global Resource Serialization (GRS) since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ Moving GRS contention notification system role
- ▶ Control the maximum number of ENQs in a system
- ▶ Changing to GRS without sysplex IPL

## 5.1 Moving GRS contention notification system role

System commands have been enhanced to help with moving the GRS contention notification role.

### 5.1.1 The issue

GRS issues an event notification facility (ENF) signal 51 to notify monitoring programs to track contention for SYSTEM and SYSTEMS scope ENQ resources. This is useful in determining contention bottlenecks, preventing bottlenecks, and potentially automating correction of these conditions.

In GRS star mode, each system only knows about the ENQs that are issued by that system. To ensure that the ENF 51 notification for SYSTEMS scope ENQs are issued on all systems in the proper sequential order, one system in the sysplex is appointed as the sysplex-wide SYSTEMS Contention Notification System (CNS). All SYSTEMS-scope ENQ contention events from all systems are sent to that system through the XCF group SYSGRS, which then issues a sysplex-wide ENF 51.

The first system in the sysplex to undergo IPL is assigned the CNS role.

In a sysplex with a lot of SYSTEMS ENQ activity, fulfilling the role of CNS can consume a considerable amount of CPU capacity. Therefore, there are some systems that you might not want to have the CNS role:

- ▶ GDPS/PPRC Control systems
- ▶ Systems with little capacity

Before z/OS 1.8, the only way to move the CNS role from the current CNS to a different system was to shut down the CNS system. The surviving systems then race to see which becomes the new CNS. You had no direct control over which system becomes the new CNS.

### 5.1.2 How this is addressed in z/OS 1.8

z/OS 1.8 introduced enhancements to two system commands:

<b>D GRS</b>	Was enhanced to display which system is the CNS
<b>SETGRS</b>	Was enhanced to add a <b>CNS=</b> parameter that lets you specify which system should be the CNS.

There is no SYS1.PARMLIB keyword to specify which system should be the CNS. However, you can issue the **SETGRS CNS=sysname** command from the console to move the CNS role to the specified system.

Alternatively, if you want, you can include the following line in your COMMNDxx member:

```
COM= 'SETGRS CNS=sysname, NP'
```

If you include this command in the IPL of every system, the following events occur:

- ▶ If the CNS is on the system that is identified on the **CNS=** parameter, it remains there.
- ▶ If the CNS is located elsewhere (perhaps because the system specified on the **CNS=** parameter underwent IPL), it is moved to that system if it is available.

### 5.1.3 Restrictions or considerations

There are no known restrictions related to the use of this function.

### 5.1.4 Rollback

APAR OA11382 rolled the ability to change the CNS system back to z/OS 1.7.

### 5.1.5 Further information

For more information about this enhancement, see the following publications:

- ▶ *z/OS MVS Planning: Global Resource Serialization*, SA23-1389
- ▶ *z/OS MVS System Commands*, SA38-0666

## 5.2 Control the maximum number of ENQs in a system

This section describes enhancements related to the maximum number of ENQs in a system.

### 5.2.1 The issue

The number of available concurrent ENQ requests has been exceeded and one of your critical tasks has failed. Before this enhancement, the only way to change these was to zap the GVTCREQA and GVTCREQ fields in the global resource serialization table (GVT).

### 5.2.2 How this is addressed in z/OS 1.8

Two new entries have been added to GRSCNFxx that allow you to specify the maximum number of concurrent authorized and non-authorized requesters. You can use the operator command `SETGRS ENQMAXA|ENQMAXU=nnnnnnnn,NOPROMPT` to alter the environment dynamically. This change will not be saved across IPLs.

### 5.2.3 Restrictions or considerations

The value for ENQMAXA can be 250,000 - 99,999,999 with a default of 250,000. The value of ENQMAXU can be 16,384 - 99,999,999 with a default of 16,384. Unless you code NOPROMPT, message ISG366D is generated and requires an acknowledgment.

### 5.2.4 Rollback

OA11382 addresses the new functions of GRS added in V1.8.

### 5.2.5 Further information

See APAR OA11382 for documentation updates before V1.8.

For information about the GRSCNFxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For a more information about GRS ENQs, see *z/OS MVS Setting Up a Sysplex*, SA23-1399.

## 5.3 Changing to GRS without sysplex IPL

### 5.3.1 The issue

If you were using another product for resource serialization, specify `GRSRNL=EXCLUDE`, indicating that GRS should not handle most serialization requests. If you then wanted to change to GRS, you must change the GRS RNLs from being excluded to using a normal full list of RNLs. However, this change might require that every system in the sysplex be shut down, and then the sysplex being brought up again with the new RNLs.

### 5.3.2 How this is addressed in z/OS 1.10

In z/OS 1.10, the GRS documentation was changed to indicate that it *is* possible to keep one system in the sysplex up while you change from `GRSRNL=EXCLUDE`. If you have critical applications that cannot afford any outage, this process might enable you to complete a change to GRS without impacting availability for those critical applications.

### 5.3.3 Restrictions or considerations

Although you can move *from* `GRSRNL=EXCLUDE` without a sysplex IPL, moving *back* to `GRSRNL=EXCLUDE` still requires a sysplex-wide IPL.

This capability can only be used in a single system sysplex in GRS STAR mode without exits `ISGNQXIT` or `ISGNQXITFAST` active. Further restrictions can be found in *z/OS MVS Setting Up a Sysplex*, SA23-1399. Careful consideration needs to be given when setting up the RNL to ensure that the proper resources are serialized.

### 5.3.4 Rollback

The ability to change from `GRSRNL=EXCLUDE` without a sysplex IPL has been available since z/OS 1.3. However, it does not work as documented before z/OS 1.10.

### 5.3.5 Further information

For more information, see *z/OS MVS Planning: Global Resource Serialization*, SA23-1389. Additional information can also be found in *z/OS MVS Initialization and Tuning Reference*, SA23-1380.



## **z/OS enhancements: Resource Recovery Services**

This chapter provides information about enhancements in z/OS RRS since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ RRS internal restart
- ▶ Clean RRS shutdown option
- ▶ Ability to dynamically turn on and off RRS archive log

## 6.1 RRS internal restart

Enhancements have been made to the Resource Recovery Services (RRS) internal restart.

### 6.1.1 The issue

Because of its sysplex nature and scope, some problems in RRS can only be resolved by canceling RRS in the entire sysplex, performing a cold start of RRS using the ATRCOLD procedure, and then restarting RRS on each system in the sysplex.

### 6.1.2 How this is addressed in z/OS 2.1

The new optional internal RRS restart is designed to quiesce RRS processing, clean up logs, and resume processing without taking RRS down. Internal Cold Start processing is designed to eliminate the sysplex-wide outage when certain problems are detected with the RM Data log. Internal Cold Start tries to resolve the problem without the outage.

Internal cold start processing, assuming there are no errors, involves this process:

- ▶ An RM DATA log problem is identified by the following message:

```
ATR212I RRS DETECTED LOG DATA LOSS ON LOGSTREAM ATR.PLEX1.RM.DATA DUE TO  
INACCESSIBLE LOG DATA. LOG DATA FROM xxx TO yyy ARE AFFECTED.
```

- ▶ The operator can decide to do an internal cold start by replying COLDSTART to message:

```
ATR250E RRS LOGSTREAM ERROR FOUND. CORRECT THE ERROR OR OPTIONALLY REPLY  
COLDSTART TO BEGIN A RRS INTERNAL COLD START.
```

After an internal cold start is started, the termination of one RRS image in the sysplex will cause all other images to terminate. The termination is identified by an MVS/RRS TERMINATION DUMP, 5C4, with REASON xxxx0029.

To Display RRS information, use this command:

```
D RRS,xxxx
```

The command supports the following parameters:

UR	Display unit of recovery information, in either summary or detailed format. You can filter the output information with the optional keyword filter parameters described below.
RM	Display resource manager information, in either summary or detailed format. You can filter the output information with the optional keyword filter parameters described below.
URSTATUS or URST	Display unit of recovery information statistics for the local system or the systems specified with the SYSNAME= and GNAME= parameters.
UREXCEPTION or UREX	Display unit of recovery information for URs that are in exceptions on the local system or the systems specified with the SYSNAME= and GNAME= parameters.

### 6.1.3 Restrictions or considerations

This ATR250E message is only issued if all the systems in the sysplex support internal cold start at the z/OS V2R1 or later level.

#### Monitor usage of RRS with Health Checker

There are multiple checks. See any exceptions for CheckOwner: **IBMRRS**.

Monitor the level of virtual storage usage in the RRS address space to prevent a terminating failure:

RRS\_Storage\_NumServerReqs

### 6.1.4 Rollback

This function is not rolled back to previous releases.

### 6.1.5 Further information

For more information about RRS, see *z/OS MVS Programming: Resource Recovery*, SA23-1395.

## 6.2 Clean RRS shutdown option

Many computer resources are so critical to a company's work that the integrity of these resources must be guaranteed. If changes to the data in the resources are corrupted by a hardware or software failure, human error, or a catastrophe, the computer must be able to restore the data. These critical resources are called **protected resources** or sometimes, **recoverable resources**.

Resource recovery consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources.

Issuing SETRRS CANCEL with non-resource manager programs in sync point might result in an X'058' abend. If the abend occurs, transactions that were in progress will be resolved when RRS restarts. Issuing SETRRS SHUTDOWN provides a normal shutdown command to bring down RRS without resulting in an X'058' abend or X'0D6' abend. To notify RRS resource managers that RRS is terminating, all the currently active resource managers are unset. After the unset processing is completed, the RRS job step task and all of its subtasks will normally be terminated to clean up the address space. In addition to the RRS infrastructure tasks, there are also timed process tasks and server tasks running in the RRS address space. These tasks are also shut down normally.

### 6.2.1 The issue

Before z/OS 1.13, there was no easy way to bring RRS down without canceling the task if any problems occurred.

## 6.2.2 How this is addressed in z/OS 1.13

RRS has been changed to allow you to request RRS to run a clean shutdown in case a unrecoverable internal error occurs. RRS issues message ATR247E to indicate that a unrecoverable internal looping problem has occurred. ATR247E has an option to allow you to request RRS to run a proper shutdown. This ensures that the proper ordering is followed while terminating RRS.

```
ATR247E RRS HAS DETECTED A SEVERE ERROR - TERMINATE RMS AND OPTIONALLY REPLY  
SHUTDOWN TO SHUTDOWN RRS
```

Usually RRS is active all the time. However, issue a SETRRS CANCEL or SETRRS SHUTDOWN command before a system IPL. Canceling RRS before an IPL results in a cleaner system recovery. This action prevents loss of data, and prevents the message ATR212I RRS DETECTED LOG DATA LOSS at IPL time.

When RRS is canceled, RRS disconnects from its log streams. When RRS disconnects, logger copies the contents of the log streams to offload data sets. This preserves the data in the RRS log streams.

## 6.2.3 Restrictions or considerations

Bring down all applications and resource managers (RMs) that use RRS services before canceling RRS. This minimizes the amount of manual intervention that is required when you restart the applications and RMs.

## 6.2.4 Rollback

This function is not rolled back to previous releases.

## 6.2.5 Further information

For more information about RRS, see *z/OS MVS Programming: Resource Recovery*, SA23-1395.

For more information about the SETRRS command, see *z/OS MVS System Commands*, SA38-0666.

# 6.3 Ability to dynamically turn on and off RRS archive log

This section covers changes to dynamically turning the RRS archive log on and off.

## 6.3.1 The issue

When RRS initializes and the RS Archive log stream has not been defined, the following message is issued:

```
ATR132I RRS LOGSTREAM CONNECT HAS FAILED FOR OPTIONAL LOGSTREAM  
ATR.##$#plex.ARCHIVE
```

Corrective action must be taken before restarting RRS.



### 6.3.2 How this is addressed in z/OS 1.10

With z/OS 1.10, you can run RRS to define the Archive log stream, then enable the log stream with the following command without having to restart RRS.

```
SETRRS ARCHIVELOGGING,ENABLE|DISABLE
```

### 6.3.3 Restrictions or considerations

Do not delete any offload data sets. This is handled by the Logger when the data expires.

To display log stream information, issue this command:

```
D  LOGGER,LOGSTREAM,LSN=ATR.*
```

#### **Monitor usage of RRS with Health Checker**

The check evaluates the coupling facility structure in which the RRS Archive log resides.

**RRS\_ArchiveCFStructure**

### 6.3.4 Rollback

This function is not rolled back to previous releases.

### 6.3.5 Further information

For more information about RRS, see *z/OS MVS Programming: Resource Recovery*, SA23-1395.

For more information about the **SETRRS** command, see *z/OS MVS System Commands*, SA22-7627.





## **z/OS enhancements: XCF and System Logger**

This chapter provides information about enhancements in z/OS Cross-System Coupling Facility (XCF) and System Logger since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ System Status Detection Partitioning Protocol
- ▶ CRITICALPAGING function for HyperSwap environments
- ▶ Non-disruptive CF dump
- ▶ XCF command to get structures to primary CF
- ▶ Force disconnection or deletion of a log stream without restarting the Logger address space
- ▶ IXGCNFxx PARM
- ▶ System Logger threshold messages
- ▶ Multiple logstream/data set management
- ▶ Dynamic log stream updates for duplexing

## 7.1 System Status Detection Partitioning Protocol

This section describes changes to the System Status Detection Partitioning Protocol.

### 7.1.1 The issue

In a Parallel Sysplex, many of the system resources are shared between the systems in the sysplex. If one of the systems stops for some reason, it is likely to be holding some resources at the instant that it stopped. It is also likely that, at some point, some work elsewhere in the sysplex will need some of those resources, and will have to wait for the resource to be released. The longer the “sick” system is left holding those resources, the larger the effect is on other systems in the sysplex. This situation is known as “sympathy sickness” because a problem on one member of the sysplex can start causing problems on other systems in the sysplex if it is not addressed.

To address this situation, IBM introduced sysplex failure management (SFM). SFM is intended to detect “sick” systems and automatically remove them from the sysplex. The mechanism for determining whether a system is sick is its heartbeat in the sysplex Couple Data Set. Normally, a system updates its heartbeat every 2 - 3 seconds. If a system stops updating its heartbeat, it is an indication to the other systems in the sysplex that this system might have a problem.

There are two reasons why a system might not update its heartbeat in the expected time:

- ▶ The system is dead.
- ▶ The system is having a problem and is trying to recover, during which time it is unable to update its heartbeat.

Consider the situation where a system is trying to recover from a problem. If it is given enough time, the recovery might be successful. So you do not want SFM to react too quickly to a missing heartbeat situation, because an IPLing of that system will take longer than waiting for recovery to work. However, if that system is dead, then there is no point waiting because it will never recover. The challenge is to find the optimum time to wait.

### 7.1.2 How this is addressed in z/OS 1.11

Starting with z/OS 1.11, XCF is able to use the z/OS BCPii function to communicate with the Support Element (SE) of the CPC that each system is running on. If the BCPii function has been customized and the sysplex couple data set (CDS) is at the correct format level, information about the LPAR is retrieved from BCPii and stored in the sysplex CDS.

If that system stops updating its heartbeat, the other members of the sysplex are able to use BCPii to get information from the SE about the *current* state of that LPAR. By comparing that information to the information that was previously placed in the sysplex CDS, XCF is able to determine these statuses:

- ▶ If that system is in a state where it might be able to recover. In that case, SFM will partition that LPAR out of the sysplex until the Failure Detection Interval has expired.
- ▶ If that LPAR is in a state that means that the previous instance of z/OS can never come back to life. In that case, SFM will immediately partition that system out of the sysplex.

This capability, which is known as System Status Detection Partitioning Protocol (SSDPP), allows SFM to remove dead systems from the sysplex in 5 to 10 seconds, compared to the 165 - 170 seconds if SSDPP is not enabled. By responding so quickly, the effect of sympathy

sickness is greatly reduced. However, the ability to detect that a system is trying to recover allows IBM to increase the default Failure Detection Interval, meaning that there is now a greater chance of that recovery completing successfully, avoiding an IPL.

To use SSDPP and BCPii, you must have the appropriate hardware and software environments, and XCF must have adequate authorization through your security product to access BCPii-related class profiles. The BCPii interface must be enabled on the hardware side with the appropriate security authorizations in place, and your sysplex CDS needs to be formatted with the SSTATDET keyword.

### 7.1.3 Restrictions or considerations

Your CPC must be at least a z10 EC GA2, a z10 BC GA1, or a later model and be running z/OS V1R11 or later to take full advantage of this capability. All systems in the sysplex must be V1R11 or have the toleration PTFs for OA26037 before the new format sysplex CDS can be used.

### 7.1.4 Rollback

Toleration of the new format sysplex CDS for pre-z/OS 1.11 systems is provided by APAR OA26037. However, the SSDPP function is not rolled back to those releases.

### 7.1.5 Further information

For more information about the SSDPP and BCPii functions, see *z/OS Setting Up a Sysplex*, SA22-7625.

For information about the COUPLEXX and EXSPATxx PARMLIB members, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more detail about BCPii, including configuration information, see *z/OS MVS Programming: Callable Services for High-Level Languages*, SA22-7612.

## 7.2 CRITICALPAGING function for HyperSwap environments

This section details enhancements to the CRITICALPAGING function used in HyperSwap environments

### 7.2.1 The issue

HyperSwap has two tasks that run on each image of the sysplex. When a HyperSwap is started, all DASD I/O are suspended, meaning that everything that is required to complete the HyperSwap operation must be in storage before the HyperSwap begins. If the HyperSwap process encounters something (some code, or a control block) that has been paged out, the HyperSwap will not be able to complete. That system will be System Reset and partitioned out of the sysplex.

For a system that does not use HyperSwap, this means tying up storage with contents that will never be used.

## 7.2.2 How this is addressed in z/OS 1.12

z/OS 1.12 includes an optional new capability known as CRITICALPAGING. If CRITICALPAGING is enabled, this significantly reduces the possibility that code or control blocks required by HyperSwap processing have been paged out. The set of declared critical system address spaces for HyperSwap are RASP (RSM), GRS, CONSOLE, XCFAS, and

the HyperSwap address spaces.

Enable **CRITICALPAGING** by updating the **COUPLExx** member with the following **CRITICALPAGING** statement on each system in the relevant sysplex.

```
FUNCTIONS ENABLE(CRITICALPAGING)
```

When the system goes through IPL, MSGIXC373I will be issued indicating that **CRITICALPAGING** is enabled.

**Restriction:** It is not possible to enable with the SETXCF command. An IPL is required.

Use this display command to see whether CRITICALPAGING is enabled:

```
D XCF,CPL
```

## 7.2.3 Restrictions or considerations

This capability applies to both Basic HyperSwap and GDPS HyperSwap, and should be enabled in any system that uses either of those functions.

Using the **SET SCH=xx** command to change the properties of a program from NOCRITICALPAGING to CRITICALPAGING does not result in the system paging in any storage associated with this program that is currently paged out. It only ensures that storage associated with the program that is currently paged in will not be paged out. Therefore, to turn on CRITICALPAGING for a specific system, update the COUPLExx member to enable CRITICALPAGING and then IPL that system. Having CRITICALPAGING enabled from IPL time is the only way to ensure that critical pages will not get paged out during the IPL.

## 7.2.4 Rollback

This function is rolled back to z/OS 1.10 and requires three APARs: OA31331(IOS), OA31707(RSM), and OA31691(XCF). All need to be active to use the CRITICALPAGING support.

## 7.2.5 Further information

For more information about the use of the **DEVSUPxx** member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the **D XCF,CPL** commands, see *z/OS MVS System Commands*, SA38-0666.

## 7.3 Non-disruptive CF dump

Enhancements have been made to non-disruptive CF dumps.

### 7.3.1 The issue

Before the z10 processors, to diagnose a coupling facility (CF) issue, you had to disruptively collect a serialized CF dump. A disruptive memory dump of a CF requires the CF to stop, dump, and then restart. All of the structures in the CF at the time of the disruptive memory dump are 'lost' when the CF restarts.

```
SETXCF DUMPCF,CFNAME=cfname,TYPE=NONDISRUPTIVE
```

The following command dumps the CF that structure is in:

```
SETXCF DUMPCF,STRNAME=strname,TYPE=NONDISRUPTIVE
```

### 7.3.2 How this is addressed in z/OS 1.13

With the z196, the IBM enhanced first-failure data capture (FFDC) introduced the ability to take a nondisruptive serialized memory dump of the CF. Because the CF is not stopped and restarted, no structures are lost.

On the CF LPAR, can issue **CFDUMP** (force a non-disruptive memory dump of the CF) if needed.

APAR OA35342 rolled back to z/OS 1.12 support of a new z/OS operator command to initiate a non-disruptive CF memory dump.

### 7.3.3 Restrictions or considerations

CF processor has to be at CFLEVEL=16 on z10 or CFLEVEL=17 on z196.

### 7.3.4 Rollback

APAR OA33723 rolled the function back to z/OS 1.9 on z10 if it is hardware supported. APAR OA31387 rolled the function back to z/OS 1.10 if on z196. Both must be in base z/OS 1.13.

### 7.3.5 Further information

For more information about the use of the **SETXCF** and **CFDUMP** commands, see *z/OS MVS System Commands*, SA38-0666.

## 7.4 XCF command to get structures to primary CF

The XCF command has been enhanced to help get structures to the primary CF.

### 7.4.1 The issue

The process to relocate structures to their wanted CF after CF maintenance or CF failure was complex and prone to operator error.

## 7.4.2 How this is addressed in z/OS 1.8

The new option **REALLOCATE** has been added to **SETXCF** command. This **SETXCF** command uses the XCF structure allocation algorithm and the structure definition in the active or pending CFRM policy to evaluate each allocated structure. When this evaluation indicates that adjustments can be made for an allocated structure, then **REALLOCATE** processing makes the appropriate adjustments.

The command **SETXCF START,REALLOCATE** starts the process.

The command **SETXCF STOP,REALLOCATE** stops the process.

When the entire process completes for all structures, the processing provides a report (message IXC545I) summarizing the actions that were taken.

## 7.4.3 Restrictions or considerations

The process of putting CF in Maintenance mode and taking it out is complex. Review the **REALLOCATE** process in the MVS “Setting up the SYSPLEX.”

## 7.4.4 Rollback

APAR OA03481 rolled **REALLOCATE** option of command back to z/OS 1.4.

With APAR OA08688 rolled back to z/OS 1.4, **REALLOCATE** eliminated needless structure rebuild processing and supported new CFRM option **ALLOWREALLOCATE** to express the action wanted when **REALLOCATE** determines that the structure instance need relocation. Avoid trying to move a structure if the application cannot handle the move.

## 7.4.5 Further information

For information about the structure **REALLOCATE** process, see *z/OS MVS Setting Up a Sysplex*, SA23-1399.

For information about the **SETXCF** and **D XCF** commands, see *z/OS MVS System Commands*, SA38-0666.

A WSC Flash about the **SETXCF START,REALLOCATE** command titled *Updated REALLOCATE Command and Coupling Facility Maintenance Procedures* is available on the IBM Techdocs website at:

<http://www.ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs>

## 7.5 Force disconnection or deletion of a log stream without restarting the Logger address space

A recent enhancement allows you to force disconnection or deletion of a log stream without restarting the Logger address space.



### 7.5.1 The issue

When trying to delete a log stream that has a failed-persistent connection, before this command, you had to cancel the IXGLOGR address space to free the connection. This process disrupts all of the tasks using logger services.

### 7.5.2 How this is addressed in z/OS 1.11

The addition of the **SETLOGR FORCE** command allows you to remove the failed connection.

To identify a problem with this log stream, issue **DISPLAY LOGGR, CONNECTION|CONN|C, LSN=logfile** and **DISPLAY LOGGR, LOGSTREAM|L, LSN=logfile** to see how many active connections there are on this system. If connections equal zero and log stream is one or more, it might be necessary to delete the log stream.

If the log stream is in a “disconnect pending” status, it might be necessary to issue a **SETLOGR FORCE, DISCONNECT, LSN=logfile** command against the log stream.

### 7.5.3 Restrictions or considerations

See **SETXCF FORCE** command to clean up coupling facility data structure resources.

If the command **SETLOGR FORCE, DELETE** has been issued, and the operation is unable to complete, the log stream is marked for deletion in the LOGR CDS. Future attempts to connect to the log stream might fail and then the delete operation is attempted.

If the **FORCE DELETE** completes successfully, but does not clean up all the resources, cleanup of staging data sets and structure connections will be attempted the next time the log stream connects. If the offload data sets are not cleaned up, they appear as “orphans” in the IXCMIAPU report and need to be manually deleted.

### 7.5.4 Further information

For more information about the SETLOGR command, see *z/OS MVS System Commands*, SA38-0666.

## 7.6 IXGCNFxx PARM

This section covers enhancements to IXGCNFxx PARM.

### 7.6.1 The issue

IXGLOGR start used default options for System Logger monitoring (in 5-second intervals) that cannot be controlled by the user. With the various IXG messages generated, there might be responses expected.

## 7.6.2 How this is addressed in z/OS 1.13

In V1.13, **IXGCNFxx** was added to specify System Logger trace options, and control structure monitoring OFFLOAD warning and action message intervals. It also defines zAware log stream client socket communication and log data buffering details to send log stream data to IBM zAware.

## 7.6.3 Restrictions or considerations

Take caution when modifying the OFFLOAD monitoring parameters, as you can define each entry from 5 seconds to 86,400 seconds (24 hours).

## 7.6.4 Rollback

This feature has not been rolled back to prior releases.

## 7.6.5 Further information

There is a description in the section “Offload and service task monitoring” in *z/OS MVS Setting Up a Sysplex*, SA23-1399 that provides a methodology for identifying delays or inhibitors to offload.

For information about **IXGCNFxx**, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

# 7.7 System Logger threshold messages

The following enhancements were made to the System Logger threshold messages.

## 7.7.1 The issue

There is a delay in a log stream offload that is not apparent until the log stream auxiliary storage is full.

## 7.7.2 How this is addressed in z/OS 2.1

You can now specify when warning messages are sent when the **HIGHOFFLOAD** value is reached/passed. Using the log stream definition provided through the administrative data utility, **IXCMIAPU**, specify **WARNPRIMARY(YES)** to increase the notification level. This causes the message **IXG317E** to alert operations when the imminent **HIGHOFFLOAD** threshold has been exceeded. Later, if the issue is not resolved, at 100%, message **IXG318E** is generated. Further control can be employed using the **IXGCNFxx** parm in **PARMLIB**.

## 7.7.3 Restrictions or considerations

Altering the definition of a log stream takes affect when the new offload data set is created.

You can specify **IXGCNF=xx** in **IEASYSxx** or use the command **SET IXGCNF=xx** to alter the definition after the system is up. This can be used to change the current state of monitoring in the event it was inadvertently turned off.

## 7.7.4 Rollback

The **IXGCNFxx** member is available in V1.13.

## 7.7.5 Further information

For a more detailed description of log stream monitoring, see *z/OS MVS Setting Up a Sysplex*, SA23-1399. There is information there on using the administrative data utility **IXCMIAPU**.

For more about **IXGCNFxx**, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

# 7.8 Multiple logstream/data set management

This section covers enhancements to multiple logstream/data set management.

## 7.8.1 The issue

Before z/OS V1R11, all staging data sets were single threaded under one task within **IXGLOGR**. This created a bottleneck.

## 7.8.2 How this is addressed in z/OS 1.11

Support has been added for separation of system logger functions between CF based and DASD-based streams. It has been designed to provide higher throughput, primary storage savings, and support higher overall concurrent offload rates. The system was modified to allow each staging data set to be allocated under the respective log stream connection task. The bottleneck is removed and allows for more concurrent allocations.

## 7.8.3 Restrictions or considerations

Production and test log stream data sets should be on separate volumes.

## 7.8.4 Rollback

This enhancement has not been rolled back.

## 7.8.5 Further information

For more information about System Logger, see *z/OS Setting Up a Sysplex*, SA22-7625.

## 7.9 Dynamic log stream updates for duplexing

Enhancements have been made for dynamic log stream updates for duplexing.

### 7.9.1 The issue

Log stream updates are placed in a pending state but not completed until a convenient time such as a data set switch, CF structure rebuild, or last connection to that log stream. The duplexing attributes (STG\_DUPLEX, DUPLEXMODE, and LOGGERDUPLEX) are only updated during the last disconnection to that log stream in the sysplex.

### 7.9.2 How this is addressed in z/OS 1.10

The change made allows you to update the variables for a particular log stream without a sysplex-wide application outage. You make the appropriate update using either the macro `IXGINVNT REQUEST=UPDATE` or using the utility program `IXCMIAPU` with `DATA TYPE(LOGR)` and `UPDATE LOGSTREAM`. To commit the update, run a user managed structure rebuild by using the `SETXCF START,REBUILD,STRNAME=structurename` command.

### 7.9.3 Restrictions or considerations

Without a V1.10 system in the SYSPLEX, the updates are committed only when the log stream is disconnected from all systems. In a mixed environment with at least one V1.10 system and other pre-V1.10 systems, the commit is completed on the first V1.10 after the structure rebuild and is used by all V1.10 systems. The pre-V1.10 systems must be disconnected from the log stream for the commit to complete.

### 7.9.4 Rollback

This has not been rolled back to previous releases.

### 7.9.5 Further information

More information can be found in the section “Updating a log stream’s attributes” in *z/OS MVS Setting Up a Sysplex*, SA23-1399.



## **z/OS enhancements: Communications Server**

This chapter provides information about enhancements in z/OS Communications Server since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS. In general, to obtain a list of Communications Server enhancements in each z/OS release, see the *z/OS Communications Server: New Functions Summary*, GC27-3664 for your z/OS release.

This chapter includes the following sections:

- ▶ Specify a hot standby server to Sysplex Distributor
- ▶ Resolver start error tolerance
- ▶ Command to verify TCP profile syntax
- ▶ RPCBIND/NFS re-registration

## 8.1 Specify a hot standby server to Sysplex Distributor

The Sysplex Distributor function balances incoming connection requests using four distribution methods and directs them to the select target TCP/IP stacks. Two of the methods use static weights and two adjust the distribution dynamically, based on periodically updated weights that are provided by Workload Manager (WLM).

### 8.1.1 The issue

Although Sysplex Distributor is primarily designed to intelligently route incoming requests across a number of server address spaces. It also allows you to route all incoming requests to a single server, and then direct all new incoming requests to a backup server. However, before z/OS 1.12, Sysplex Distributor did not support having one active target with multiple backups.

### 8.1.2 How this is addressed in z/OS 1.12

The Communications Server component of z/OS 1.12 introduced Sysplex Distributor support for hot-standby servers by using a new distribution method called *Hot-Standby*. To use this capability, configure a preferred server and one or more hot-standby servers. The preferred server has an active listener and receives all new incoming connection requests, and the hot-standby servers act as backup servers in case the designated preferred server becomes unavailable. You can rank the hot-standby servers to control which hot-standby server becomes the active server. You can also control whether the Sysplex Distributor automatically switches back to using the preferred server if it again becomes available, and whether the distributor automatically switches servers if the active target is not working correctly.

Configure the hot-standby function by specifying the following parameters and options on the VIPADISTRIBUTE statement:

- ▶ The **HOTSTANDBY** option on the **DISTMETHOD** parameter
- ▶ Optionally, the **NOAUTOSWITCHBACK** and **NOHEALTHSWITCH** options for **HOTSTANDBY** on the **DISTMETHOD** parameter
- ▶ The **PREFERRED** or **BACKUP** option on the **DESTIP** parameter

**Note:** **PREFERRED** or **BACKUP** must be configured on the **DESTIP** parameter for each target after the dynamic cross-system coupling facility (XCF) address. For more information, see the description of the **DESTIP HOTSTANDBY** options.

**IPCONFIG SYSPLEXROUTING** must be specified on all target systems for this distribution method to be used.

### 8.1.3 Restrictions or considerations

The Sysplex Distributor must be at the z/OS 1.12 or later level, and the TCP/IP target stacks must be at the z/OS 1.10 or later level. There are also some configuration restrictions:

- ▶ You cannot configure **DESTIP ALL** with this distribution method.
- ▶ You cannot configure **GRE** or **ENCAP** with this distribution method.
- ▶ **TIMEDAFFINITY** is ignored if you use this distribution method.

## 8.1.4 Rollback

This function is not rolled back to previous releases.

## 8.1.5 Further information

For more information about the **VIPADISTRIBUTE** statement, see *z/OS IBM Communications Server: IP Configuration Reference*, SC27-3651.

For more information about the configuring the hot standby function, see *z/OS IBM Communications Server: IP Configuration Guide*, SC27-3650.

## 8.2 Resolver start error tolerance

The system resolver was first incorporated into z/OS Communications Server in z/OS 1.2. Its purpose was to simplify definition and management by reducing the number of separate resolvers in the product (although some applications, such as SMTP, still maintain their own separate resolver).

The resolver setup file was defined at the same time the system resolver was created. The setup file provides an optional mechanism for specifying configuration information for overall resolver processing. Originally, there were only two configuration statements supported: GLOBALTCPDATA and DEFAULTTCPDATA.

### 8.2.1 The issue

Over time, more configuration statements were added and the complexity of setting up system resolver grew. Additionally, this was complicated by some configuration statements not being recognized on all z/OS releases, making sharing difficult between systems. Throughout the addition of new options, the original philosophy remained in resolver: Fail upon detection of any setup file error. Moreover, failure of resolver to initialize would cause failure of the TCP/IP stack to initialize, meaning that dependent applications also could not start.

### 8.2.2 How this is addressed in z/OS 2.1

To make resolver more forgiving of errors in its setup file, and to ensure that any such errors do not stop TCP/IP from initializing, the following changes were made to resolver initialization processing in z/OS 2.1:

- ▶ If a valid statement has a parameter error, a warning message is displayed on the operator console and in the JES job log.
  - If the statement was found during resolver address space initialization, the statement is ignored. Processing of the setup statements continues.
  - If the statement was found while processing a **MODIFY RESOLVER, REFRESH, SETUP** command, processing of the setup statements ends, the **MODIFY** fails, and no refresh takes place.

- ▶ If an invalid statement is found, a warning message is displayed on the operator's console and in the JES job log.
  - If the invalid statement was found during resolver address space initialization, the statement is ignored. Processing of the setup statements continues.
  - If the invalid statement was found while processing a **MODIFY RESOLVER, REFRESH, SETUP** command, processing of the setup statements ends, the **MODIFY** fails, and no refresh takes place.

When setup file processing is complete, the resolver attempts to open the MVS data set or z/OS UNIX file name specified as the parameter on the last valid instance of the GLOBALIPNODES, DEFAULTIPNODES, GLOBALTCPIPDATA, and DEFAULTTCPIPDATA setup statements. If the parameter value was incorrect or the specified z/OS UNIX or MVS data set does not exist, a warning message is displayed on the operator console and in the JES job log:

- ▶ If the error occurred during resolver address space initialization, processing continues. The resolver assumes the default setting for the setup statement, which in this case is no file was specified.
- ▶ If the error occurred while processing a **MODIFY RESOLVER, REFRESH, SETUP** command, processing of the setup statements ends, the **MODIFY** fails, and no refresh takes place.

If an allocation error occurs when trying to access a resolver statement's MVS data set or z/OS UNIX file, an eventual action message is issued to the operator console. Only one eventual action message per MVS data set or z/OS UNIX file is issued regardless of the number of times the file is accessed. After a successful reference to the file occurs, the message is removed from the operator console.

- ▶ If the allocation error occurs during resolver address space initialization, processing continues.
- ▶ If the allocation error occurs while resolver is processing a **MODIFY RESOLVER, REFRESH, SETUP** command, processing of the setup statements ends and the **MODIFY** command fails. No refresh takes place.
- ▶ If the allocation error occurs during a resolver API call, processing of the resolver API continues. The resolver takes default values, if appropriate.

If no errors are detected during resolver address space initialization, the resolver issues message EZZ9291I at the completion of initialization processing. If an error is detected, the resolver issues message EZD2038I. You can create automation to monitor for message EZD2038I and to alert you about errors in your resolver setup file that might cause the resolver to behave differently than intended.

To determine the current setting of the resolver setup statements, use the following command:

```
MODIFY RESOLVER,DISPLAY
```

## 8.2.3 Restrictions or considerations

If resolver setup statements are contained in a z/OS UNIX file, the file can have a maximum line length of 256.

## 8.2.4 Rollback

This function is not rolled back to previous releases.



## 8.2.5 Further information

For more information about the **RESOLVER** start, see *z/OS IBM Communications Server: IP Configuration Reference*, SC27-3651.

## 8.3 Command to verify TCP profile syntax

A TCP/IP profile is a data set or collection of data sets that contain statements to configure a TCP/IP stack. The stack reads the profile and processes the configuration statements when you start the stack. The stack also reads and processes a profile when you issue the **VARY TCPIP,,OBEYFILE,profile** command to change the configuration, where **profile** is the profile you want to activate.

### 8.3.1 The issue

The TCP/IP profile parser makes a single pass of the TCP/IP profile, reading each statement and saving it before parsing the next statement. If a syntax error is encountered, it either discards the statement, or overrides the values that are coded with the defaults. At the end of parsing, information is processed by the TCP/IP stack.

Any change in TCP/IP profiles is not validated until TCP/IP is recycled or the system goes through IPL. Problems in the profile might lead to unpredictable or undesirable results, a partial profile activation, or even an unintended outage. To recover, you might need to correct it from another system or use a backout member to get TCP/IP to work.

### 8.3.2 How this is addressed in z/OS 2.1

The availability of TCP/IP is improved by providing a method to check the syntax of TCP/IP profile statements in an initial profile or in the profile data set that is specified on a **VARY TCPIP,,OBEYFILE** command without activating the profile. With the **VARY TCPIP,,SYNTAXCHECK** command, you can check the syntax of configuration statements in profile data sets before using the statements to configure TCP/IP.

You do not need to issue the command on the system that will apply the profile. You can check the profile on any system that supports the **VARY TCPIP,,SYNTAXCHECK** command. For example, you can specify a TCP/IP stack on this command that is configured to support only IPv4 to check a profile that contains IPv6 profile statements.

### 8.3.3 Restrictions or considerations

Issue the **VARY TCPIP,,SYNTAXCHECK** command on the same z/OS release as that of the target TCP/IP stack. This ensures consistent syntax checking.

Issue the command repeatedly, if necessary, to verify that the TCP/IP profile is free of syntax errors before activating the profile. Sometimes the TCP/IP stack stops processing when it finds an error. Therefore, by having a clean validation of the TCP/IP profile, you can verify that the entire profile is checked.

### 8.3.4 Rollback

This function is not rolled back to previous releases.

### 8.3.5 Further information

For more information about the **VARY TCPIP, ,SYNTAXCHECK** command, see *z/OS IBM Communications Server: IP System Administrator's Commands*, SC27-3661.

## 8.4 RPCBIND/NFS re-registration

RPCBIND/NFS re-registration has been enhanced.

### 8.4.1 The issue

RPCBIND and NFS Servers have these characteristics:

- ▶ Designed to allow the NFS Server to reregister with RPCBIND when RPCBIND is restarted
- ▶ Designed to help preserve existing connections
- ▶ Designed to allow new mounts when RPCBIND is restarted
- ▶ Intended to let you avoid an NFS Server restart to improve availability

### 8.4.2 How this is addressed in z/OS 2.1

The rpcbind server is improved to provide notifications at strategic points in processing and to enable more effective programming. The rpcbind server sends an Event Notification Facility (ENF) signal when the server is starting and when it is stopping.

- ▶ The rpcbind server sends an ENF signal when it starts and is prepared to accept registrations from Remote Procedure Call (RPC) applications. If the rpcbind server is stopped and restarted, RPC applications can monitor this ENF signal and register again with the rpcbind server.
- ▶ The rpcbind server sends an ENF signal when it is stopped or canceled. If the rpcbind server is not available to RPC clients, RPC applications can monitor this ENF signal and take action.

### 8.4.3 Restrictions or considerations

There are no known restrictions or considerations.

### 8.4.4 Rollback

This function is not rolled back to previous releases.

### 8.4.5 Further information

For more information about the **RPCBIND** command, see *z/OS IBM Communications Server: IP System Administrator's Commands*, SC27-3661.



## z/OS enhancements: DFSMS

This chapter provides information about enhancements in z/OS DFSMS since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ Dynamically activate changes to DEVSUPxx member
- ▶ Dynamically modify size of SMSPDSE hiperspaces
- ▶ VSAM CA reclaim for KSDSs
- ▶ Support for very large catalogs and catalogs on extended address volumes (EAVs)
- ▶ Minimizing application disruption during catalog maintenance
- ▶ DADSM and CVAF support for dynamic corrective service activation
- ▶ DFSMSHsm Control Data Set backup enhancements
- ▶ Allow XRC primary volumes to be offline when the XSTART and XADDPAIR commands are issued
- ▶ Metro Mirror (PPRC) support for Remote Pair FlashCopy
- ▶ Remove need to offline/online volume after VTOC change
- ▶ Dynamically update maximum number of tape drives OAM allocates

## 9.1 Dynamically activate changes to DEVSUPxx member

Enhancements have been made to help dynamically activate changes to DEVSUPxx members.

### 9.1.1 The issue

The DEVSUPxx member of PARMLIB specifies the installation defaults for many tape-related device support options, and some related to DASD. It controls options such as the use of compaction, system-determined blocksize, tape security, and others.

The DEVSUPxx member is processed during the NIP phase of IPL. All of these options can be overridden using JCL keywords, and the **DEVSERV QL,xxx** command can be used to change the default value of a subset of the keywords. However, before z/OS 1.8, the only way to activate an updated DEVSUPxx member was to perform an IPL.

### 9.1.2 How this is addressed in z/OS 1.8

z/OS 1.8 added a command, **SET DEVSUP=xx**, that lets you dynamically activate a changed DEVSUPxx member. The scope of this command is a single system, so if you want to implement the change across every member of a sysplex, you must issue this command on every system.

There is no command that will display the current values of all the system settings that are controlled by the DEVSUPxx member.

### 9.1.3 How this is addressed in z/OS 1.10

z/OS 1.10 added the **CATS** option to the **DEVSERV QL,xxx** command to display or dynamically change the category codes for tape library partitioning. APAR OA24965 rolled this capability back to z/OS 1.8. Note that the scope of this command is also a single system, so if you want to implement the change across every member of a sysplex, you must remember to issue this command on every system.

### 9.1.4 Restrictions or considerations

In z/OS 1.8, the **SET DEVSUP=xx** command could not be used to change the category codes for Tape Library Partitioning dynamically. However, this restriction was removed in z/OS 1.12.

### 9.1.5 Rollback

The **SET DEVSUP=xx** command is only available on z/OS 1.8 and later systems. The **DEVSERV QL** command to display and change the category codes (CATS) was rolled back to z/OS 1.8.

### 9.1.6 Further information

For more information about the use of the **SET DEVSUP=xx** and **DEVSERV** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about the use of the DEVSUPxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 9.2 Dynamically modify size of SMSPDSE hiperspaces

The following enhancements were made to help dynamically modify size of SMSPDSE hiperspaces.

### 9.2.1 The issue

In z/OS releases before z/OS 1.8, the only way to change the size of the hiperspace used by the SMSPDSE1 address space was to restart SMSPDSE1.

### 9.2.2 How this is addressed in z/OS 1.8

A new parameter on the **SETSMS** command, **PDSE1\_HSP\_SIZE**, allows the hiperspace size to be altered dynamically. Remember to update the **PDSE1\_HSP\_SIZE** parameter in the IGDSMSxx member so that the change is not regressed when the system goes through IPL.

The command **D SMS,OPTIONS** shows the contents of the IGDSMSxx member that was used for the most recent IPL or **SET SMS=xx** command.

**Note:** When you issue the **SETSMS PDSE1\_HSP\_SIZE(xx)** command, you will not receive any response. If you issue a **D SMS,OPTIONS** command, you see that the command *was* accepted. However, the change will not take effect until the SMSPDSE1 address space is restarted.

### 9.2.3 Restrictions or considerations

This capability is only available for the SMSPDSE1 address space because the address space must be restarted for the change to take effect, and it is not possible to restart the SMSPDSE address space.

### 9.2.4 Rollback

This capability is not rolled back to previous releases.

### 9.2.5 Further information

For more information about the **SETSMS** and **D SMS** commands, see *z/OS MVS System Commands*, SA38-0666.

For more information about the IGDSMSxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

## 9.3 VSAM CA reclaim for KSDSs

This section covers enhancements to the VSAM CA reclaim for key-sequenced data sets (KSDSs).

### 9.3.1 The issue

VSAM KSDSs with many deletes end up having many empty control areas (CAs). The only way to make those empty CAs available for use again is to reorganize the data set, meaning that no other jobs or started tasks can use it during that time.

### 9.3.2 How this is addressed in z/OS 1.12

A new CA reclaim function delivered in z/OS 1.12 lets you specify that empty CA space is to be reclaimed automatically. CA reclaim applies to VSAM KSDSs, alternate indexes (AIXs), and catalogs. There is no CA reclaim processing for ESDS, RRDS, VRRDS, or LINEAR data sets. It works for both SMS-managed and no-SMS-managed data sets. This eliminates the outage that is associated with the reorganization activity, and can also reduce fragmented DASD space and data sets growing to many extents. However, CA reclaim does not eliminate the need to perform data set reorganizations, but it should reduce the frequency at which you need to reorganize the data set.

CA reclaim is disabled at the system level by default. To enable CA reclaim for the system, use one of these methods:

- ▶ Specify `CA_RECLAIM(DATACLAS)` in the `IGDSMSxx` member of `PARMLIB` and then IPL the system
- ▶ Issue the **SETSMS CA\_RECLAIM(DATACLAS)** command

After CA reclaim is enabled for the system, the following are true:

- ▶ It is in effect for existing KSDSs.
- ▶ When new data sets are defined, they use the setting for the CA reclaim attribute in the data class. By default, the setting is Yes, enabling CA reclaim (if it is enabled at the system level). You can change the setting to No, so that when new data sets are defined using that data class, CA reclaim is disabled.
- ▶ You can disable CA reclaim for individual data sets with the **IDCAMS ALTER RECLAIMCA** command.

CA reclaim will not reclaim these items:

- ▶ Partially empty CAs.
- ▶ A CA that was already empty before CA reclaim is enabled.

CAs that were empty before CA reclaim is enabled will be reclaimed the next time the data set is reorganized.

- ▶ CAs in KSDSs with IMBED.
- ▶ Application opening a data set with global shared resources (GSRs).

**Note:** The **SETSMS** command and the `IGDSMSxx` member only control the state of the CA reclaim function for one system. If you want to enable it on all systems in the sysplex, use the **R0 \*ALL** command to route the **SETSMS** command to all systems in the sysplex.

You can display the CA reclaim attribute for a VSAM data set using the IDCAMS **LISTCAT** command. To determine whether CA reclaim is desirable for a data set, use the IDCAMS **EXAMINE DATASET** command. **EXAMINE DATASET** shows the number of empty CAs in a KSDS with message IDC01728I. There is also a tool available that produces a one line listing of data sets that are currently eligible for CA RECLAIM. This produces a more concise report than using the IDCAMS LISTCAT command. This tool can be downloaded from this FTP site:

<ftp://public.dhe.ibm.com/servers/storage/support/software/dfsms/>

It is on this site as CARECLAM.JCL.CNTL.TRSD.

### 9.3.3 Restrictions or considerations

If you encounter any major issues, you can turn CA reclaim off on a system-by-system basis by using the following command:

```
SETSMS CA_RECLAIM(NONE)
```

Remember to update the IGDSMSxx member so that it does not get turned back on at the next IPL.

### 9.3.4 Rollback

This function was not rolled back to previous releases.

### 9.3.5 Further information

For more information about the use of the IGDSMSxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the use of the **SETSMS** command, see *z/OS MVS System Commands*, SA38-0666.

For more information about the CA reclaim function, see *z/OS DFSMS Using Data Sets*, SC23-6855.

For up-to-date information about this function, see Information APAR II14640.

## 9.4 Support for very large catalogs and catalogs on extended address volumes (EAVs)

The following enhancements were made concerning support for very large catalogs and catalogs on extended address volumes (EAVs).

### 9.4.1 The issue

Before z/OS 1.12, ICF catalogs could not be larger than 4 GB. If a catalog filled, moving catalog entries to a new catalog could be disruptive. Similarly, it was disruptive if a catalog ran out of space and needed to be reorganized.

## 9.4.2 How this is addressed in z/OS 1.12

Starting with z/OS 1.12, catalogs and more types of data sets became external authorization service (EAS)-eligible with the implementation of extended address volumes (EAVs). This enhancement allows catalogs to grow beyond the previous 4-GB size limit using extended addressability (EA). A catalog that is defined to use extended addressability is known as an Extended Attribute Catalog. To change to an Extended Attribute Catalog requires updating the DATACLASS, which cannot be altered. Therefore, complete the steps documented in the chapter “Alter Catalog Attributes” in *z/OS DFSMS Managing Catalogs*, SC26-7409.

Because ICF catalogs are EAS-eligible, they can be placed anywhere on an EAV.

**Note:** See 3.4.1, “Extended Address Volume (EAV)” on page 33 for more detail about EAV volumes and EAS-eligibility.

## 9.4.3 Restrictions or considerations

Unlike other extended format VSAM data sets, which can also be striped or compressed, extended format ICF catalogs cannot be striped or compressed. Also, this catalog enhancement does *not* apply to VSAM volume data sets (VVDs), which continue to have a 4-GB size limit.

## 9.4.4 Rollback

This function is not rolled back to previous releases.

## 9.4.5 Further information

For more information about how to set up and use EAV, see *z/OS DFSMS Using Data Sets*, SC23-6855 and *DFSMS V1.10 and EAV Technical Guide*, SG24-7617.

For more information about the use of Extended Attribute Catalogs, see *z/OS DFSMS Managing Catalogs*, SC26-7409.

# 9.5 Minimizing application disruption during catalog maintenance

The enhancements described in this section help you minimize application disruption during catalog maintenance.

## 9.5.1 The issue

Any outage that affects an ICF catalog can be disruptive. During the time of the outage, data sets defined in that catalog cannot be accessed unless the volser of the data set is known (it normally is not). And even if the volser is known, if the data set is a VSAM data set or is SMS-managed, it cannot be accessed if the catalog is unavailable.

However, there are some situations where a catalog must be made unavailable, such as to address a problem, or for maintenance. Before z/OS 2.1, any jobs or started tasks that attempted to use that catalog to get data set information would fail.



## 9.5.2 How this is addressed in z/OS 2.1

z/OS 2.1 provides two new commands to suspend/resume requests to a catalog during catalog recovery or maintenance.

The **F CATALOG,RECOVER,SUSPEND(ucatname\*)** command will quiesce catalog requests and close the catalog across the sysplex.

When you have finished your catalog activity, use the **F CATALOG,RECOVER,RESUME(ucatname\*)** command open the catalog and resume requests across the sysplex.

Two new commands for RLS to quiesce/enable the RLS catalog's request. Catalog requests are suspended during the process of enabling or quiescing RLS access for catalogs. No catalog requests will fail as a result of the enable or quiesce.

The following is the command for sysplex-wide switch to non-RLS. It closes catalog requests for RLS, and is opened for non-RLS at the next non-RLS access.

```
F CATALOG,RLSQUIESCE(ucat)
```

The following is the command for sysplex-wide switch to RLS. It opens the catalog for RLS request at the next RLS access.

```
F CATALOG,RLSENABLE(ucat)
```

**Note:** Issuing the **RLSENABLE** command cannot cancel or reverse a running **RLSQUIESCE** command or vice versa. Each command must either fully complete or time out before another command can take effect.

## 9.5.3 Restrictions or considerations

There are no known restrictions related to the use of this function.

## 9.5.4 Rollback

This function was not rolled back to previous releases.

## 9.5.5 Further information

For more information about managing catalog access during recovery or maintenance activity, see the section titled "Locking a Catalog" in the z/OS 2.1 version of *z/OS DFSMS Managing Catalogs*, SC26-7409.

## 9.6 DADSM and CVAF support for dynamic corrective service activation

The ability to update load modules in LPA (known as Dynamic LPA) has been available for some time, but not all components supported it. It is an attractive capability because it allows you to install critical service that updates a module in LPA without requiring an IPL.

For components that use Dynamic LPA, the Dynamic LPA exit (CSVDYLPA) notifies them when modules are added or deleted from LPA. By using this notification, any internal control structures can be updated as necessary.

For more information, see the section titled “Dynamic LPA” in *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

### **9.6.1 The issue**

DADSM (direct access device space management) and CVAF (command VTOC access facility) are two important components in DFSMS that can have critical services that require an IPL to implement.

### **9.6.2 How this is addressed in z/OS 1.13**

Starting with z/OS 1.13, PTFs to address critical problems in DADSM or CVAF that update LPA can now potentially be applied without an IPL. Any PTFs that use this capability must contain the HOLD for DYNACT, providing instructions about how to activate the service.

### **9.6.3 Restrictions or considerations**

Using Dynamic LPA to avoid an IPL for CVAF and DADSM is most likely to be possible if you are reasonably up to date in service. If the PTF you are applying has a long prerequisite chain (most likely to be the case if you have not applied service for a long time), some of the prerequisites might require an IPL. If that is the case, it is not possible to use the Dynamic LPA capability.

### **9.6.4 Rollback**

This function is not rolled back to previous releases.

### **9.6.5 Further information**

For more information about Dynamic LPA, see the section titled “Dynamic LPA” in *z/OS Planned Outage Avoidance Checklist*, SG24-7328.

## **9.7 DFSMSHsm Control Data Set backup enhancements**

The following enhancements have been made for DFSMSHsm Control Data Set backup.

### **9.7.1 The issue**

Many clients run frequent backups of the DFSMSHsm control data sets (CDSs) to minimize recovery time in case one of those data sets gets lost or damaged. However, the backup process requires serialization of the CDSs, meaning that the backup is disruptive to other DFSMSHsm processes. Depending on the elapsed time for the backup, this can have an impact on other jobs. For example, RECALL requests for data sets required by production jobs cannot be processed during the backup window.

## 9.7.2 How this is addressed in z/OS 1.13

Starting with z/OS 1.13, DFSMSHsm CDS backup processing is enhanced when you specify that a point-in-time copy technique is to be used. The backup process now begins backing up the CDSs immediately instead of waiting for DFSMSHsm requests to complete, and allows DFSMSHsm to continue processing requests with only brief interruptions. This avoids suspending DFSMSHsm request during the CDS backup.

Additionally, the journal backup process has been enhanced through the introduction of non-intrusive journal backup. During non-intrusive journal backup, all DFSMSHsm activity is allowed to continue while DFSMSHsm creates a backup of the existing journal records. After the initial backup of the journal completes, all DFSMSHsm activity is quiesced and DFSMSHsm completes the backup of the journal and backs up the control data sets. Although the overall elapsed time for non-intrusive journal backup is approximately the same as quiesced journal backup, non-intrusive journal backup can be used to reduce the impact on DFSMSHsm availability and performance

## 9.7.3 Restrictions or considerations

There are no known restrictions related to the use of this function.

## 9.7.4 Rollback

This function is not rolled back to previous releases.

## 9.7.5 Further information

For more information about DFSMSHsm backups, see *z/OS DFSMSHsm Storage Administration Reference*, SC35-0421.

# 9.8 Allow XRC primary volumes to be offline when the XSTART and XADDPAIR commands are issued

Thanks to recent enhancements, XRC primary volumes can be offline when the XSTART and XADDPAIR commands are issued.

## 9.8.1 The issue

The current implementation of XRC requires that primary volumes (the volumes being written to by application programs) are online to the XRC system when the volumes are added to an XRC session, and also when an XRC session is restarted. As configurations have grown from hundreds, to thousands, to tens of thousands of primary volumes, the time to vary these volumes online to the XRC system has become very large.

If the XADDPAIR command is issued for a volume that has not yet been varied online to the XRC system, the XADDPAIR processing fails. All failed XADDPAIR commands then must be redone after identifying the failures.

## 9.8.2 How this is addressed in z/OS 2.1

z/OS 2.1 addresses this issue by allowing XRC primary volumes to be offline when the XSTART and XADDPAIR commands are issued to start or restart mirroring for existing volumes.

This is enabled by a new function called REDISCOVER. Issuing a **F ANTAS000,REDISCOVER** command forces a rediscovery of offline devices on the XRC LPAR. This might be useful following a dynamic I/O reconfiguration when rediscovery can shorten the time that subsequent XRC commands take to process. It is also useful after an IPL of the XRC system, when you want to restart mirroring as quickly as possible.

## 9.8.3 Restrictions or considerations

The **F ANTAS000,REDISCOVER** command can be issued only from a system that has the ANTAS000 address space active.

## 9.8.4 Rollback

This enhancement is rolled back to z/OS 1.10 and later releases by APAR OA36570.

## 9.8.5 Further information

For more information about XRC and the REDISCOVER function, see *z/OS DFSMS Advanced Copy Services*, SC35-0428.

# 9.9 Metro Mirror (PPRC) support for Remote Pair FlashCopy

This section covers enhancements to the Metro Mirror support for IBM Remote Pair FlashCopy®.

## 9.9.1 The issue

The FlashCopy function is used to take a point-in-time copy of a volume or a data set. Metro Mirror (also known as PPRC) is used to maintain a mirror copy of a volume, typically for disaster recovery reasons. To maintain disaster recover readiness, you want all PPRC pairs to constantly be in a full duplex state, meaning that the primary and secondary volumes are identical.

Before z/OS 1.11, if you initiate a FlashCopy operation from one primary device to another, the state of the PPRC pair for the target volume would change to PENDING until the FlashCopy completes, and the primary and secondary PPRC volumes are fully synchronized again.

The relationship is shown in Figure 9-1.

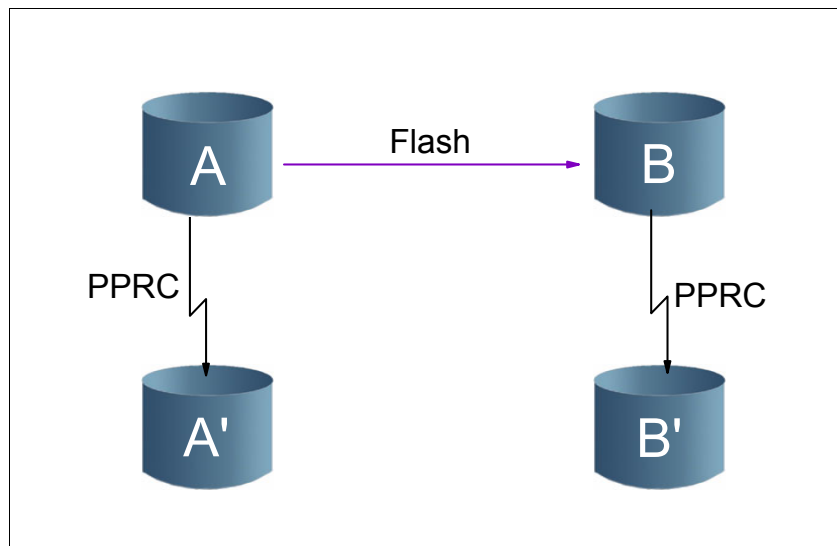


Figure 9-1 Combining FlashCopy and PPRC without Preserve Mirror

### 9.9.2 How this is addressed in z/OS 1.11

z/OS 1.11, together with support in the DASD subsystem, introduced an option called Preserve Mirror. When Preserve Mirror is used, a FlashCopy is taken from the primary volume A (Figure 9-2) to volume B, a FlashCopy is taken from A' to B', and the PPRC relationship between volume B and B' remains in the full duplex state.

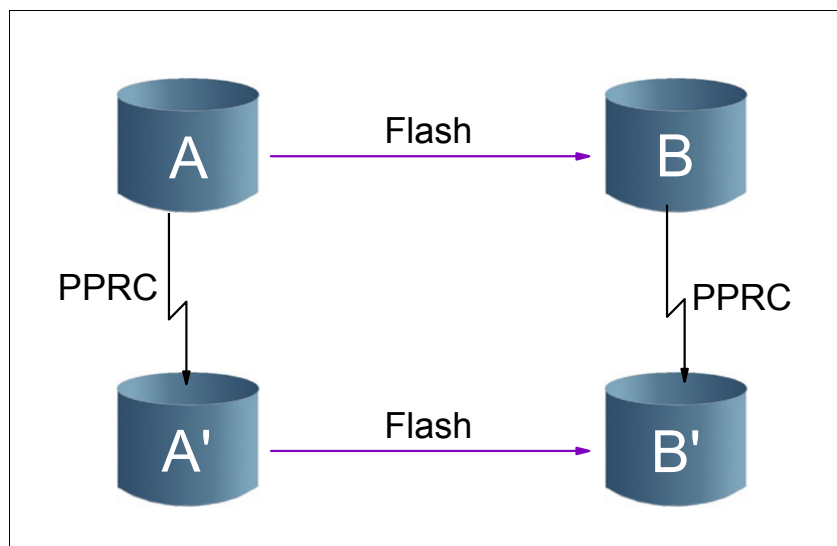


Figure 9-2 Combining FlashCopy and PPRC with Preserve Mirror

During the FlashCopy operation, no data is sent from volume B to B'. Rather, the data is copied from A' to B'.

The following are the requirements to be able to use Preserve Mirror:

- ▶ Both volumes A and B must be PPRC primary devices, in full duplex mode with links established and available.
- ▶ Volumes A' and B' must be in the same storage facility image (SFI).
- ▶ Volumes A and B can be the same device for an operation at the data set level.
- ▶ The required microcode level must be installed in both the local and the remote storage control units.

### 9.9.3 Restrictions or considerations

The physical DASD CU must be at the supported level of microcode to support this function.

### 9.9.4 Rollback

APAR OA24811 rolled the new function, Preserve Mirror FlashCopy, back to z/OS 1.8.

### 9.9.5 Further information

For more information about using FlashCopy and Metro Mirror together, see *z/OS DFSMS Advanced Copy Services*, SC35-0428.

## 9.10 Remove need to offline/online volume after VTOC change

Recent enhancements have removed the need to offline/online a volume after a VTOC change.

### 9.10.1 The issue

Before z/OS 1.13, any time a VTOC was updated by a DFSMSdss COPY or RESTORE, or by an ICKDSF REFORMAT NEWVTOC operation, the volume had to be taken offline and online again on every system that had access to it. This was required to update the UCB for that volume.

### 9.10.2 How this is addressed in z/OS 1.13

z/OS 1.13 delivered an enhancement to the Device Manager (DEVMAN) whereby it will automatically refresh the UCB after a DFSMSdss COPY or RESTORE operation or an ICKDSF REFORMAT NEWVTOC operation. Those operations might cause the volume serial and the VTOC location on the target volume to change.

If the device is ONLINE, and the REFUCB function is enabled, DEVMAN automatically issues a **VARY ONLINE, UNCONDITIONAL** command. This updates both the volser and VTOC location in the UCB. If the device is OFFLINE, no action is taken. This processing takes place on every system that is in the same sysplex as the system that updated the volume.

To enable this capability, you need to update the DEVSUPxx member of PARMLIB and enable REFUCB. You can also enable it dynamically using the **F DEVMAN,ENABLE(REFUCB)** command.

To check the status of this function, use the **F DEVMAN,REPORT** command, which shows the current setting of the REFUCB keyword as shown in Example 9-1.

*Example 9-1 Displaying status of REFUCB function*

---

```
F DEVMAN,REPORT
DM00030I DEVICE MANAGER REPORT 258
**** DEVMAN ****
* FMID: HDZ2210 *
* APARS: NONE *
* OPTIONS: REFUCB *
* HPF FEATURES DISABLED: NONE *
* NO SUBTASKS ARE ACTIVE *
**** DEVMAN ****
```

---

### 9.10.3 Restrictions or considerations

Because DEVMAN uses XCF to communicate to its peers, the ability to automatically update the UCB is limited to systems in the same sysplex as the system that updated the VTOC. Do not share volumes across multiple sysplexes. If you do, then you are still responsible for updating the UCB on systems in those other sysplexes.

If REFUCB is implemented dynamically, remember to update the DEVSUPxx member so that it does not get regressed with the next IPL.

### 9.10.4 Rollback

This function was not rolled back to previous releases.

### 9.10.5 Further information

For more information about the **F DEVMAN** command, see the section titled “Communicating with the Device Manager Address Space” in *z/OS MVS System Commands*, SA38-0666.

For more information about the DEVSUPxx member, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

For more information about the REFUCB function, see *z/OS Version 2 Release 1 Implementation*, SG24-xxxx.

## 9.11 Dynamically update maximum number of tape drives OAM allocates

Enhancements now allow you to dynamically update the maximum number of tape drives in Object Access Method (OAM) without requiring a restart.

### 9.11.1 The issue

The CBROAMxx PARMLIB member is used to control the OAM settings. The CBROAMxx member is processed during OAM address space initialization.

Before z/OS 1.13, changing the maximum number of tape drives required editing the SGMXTAPESTORETASKS and SGMXTAPERETRIEVETASKS values in the CBROAMxx PARMLIB member and restarting OAM.

### 9.11.2 How this is addressed in z/OS 1.13

In z/OS 1.13, the **F OAM,UPDATE,SETOAM** command has been enhanced to let you change the SGMXTAPESTORETASKS and SGMXTAPERETRIEVETASKS settings without having to restart OAM. Use the **F OAM,DISPLAY,xxxxxxx,ALL** command to display the current settings before you make any changes, and again after the change to ensure that it was successful.

### 9.11.3 Restrictions or considerations

If you restart OAM and no specific statements exist in the CBROAMxx PARMLIB member for a specific storage group and a previous UPDATE SETOPT, SETOSMC, or SETOAM command was issued to update a field for that storage group, the value assigned by the update might be retained.

To check whether you can update a setting dynamically, see the section titled “Using the UPDATE command to set SETOAM, SETOSMC, and SETOPT” in *z/OS V1R13 DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*, SC35-0426.

### 9.11.4 Rollback

This function is not rolled back to previous releases.

### 9.11.5 Further information

For more information about the CBROAMxx member, see *z/OS V1R13 DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*, SC35-0426.





## z/OS enhancements: Distributed File Service (zFS)

This chapter provides information about Distributed File Service (zFS) enhancements in z/OS since z/OS 1.8 that help you avoid or postpone a planned outage.

This chapter includes the following sections:

- ▶ zFS internal restart
- ▶ zFS automatic re-enable of disabled file systems

**Note:** The direction that IBM is taking with zFS is that there will be a one-to-one relationship between aggregates and file systems. In the past, this was not always the case, so the differentiation between “file system” and “aggregate” was more important. However, to simplify the text, and to more accurately reflect the future situation, this section uses the term “file system” exclusively. However, some zFS console messages still use the term “aggregate.” Within the context of this book, consider aggregate to be synonymous with file system.

## 10.1 zFS internal restart

From the time z/OS 1.7 was released, clients have been moving from HFS to zFS file systems. zFS has been incorporated into several enhancements in the area of resiliency, which provides higher availability for the zFS file systems.

### 10.1.1 The issue

Before z/OS 1.13, if zFS encountered an internal problem, it would take a memory dump. Then, depending on what had happened, zFS might perform one of these actions:

- ▶ Decide to continue processing
- ▶ Disable a zFS file system
- ▶ Stop and request that z/OS UNIX restart zFS.

The shortcoming was that if zFS decided to stop and request a restart, zFS file systems might be unmounted based on what happened. For example, a stop and restart would be done for physical file system (PFS) layer errors or exceptions.) Unmounting file systems means that critical data and programs would be unavailable.

### 10.1.2 How this is addressed in z/OS 1.13

As of z/OS 1.13, in situations where zFS would have stopped itself and requested that z/OS UNIX restart it, zFS now does an internal restart instead. The benefit of this internal restart is that the restart time can be reduced, reducing zFS file system unavailability.

There is no explicit action that is required by the client to get this new behavior. zFS runs the internal restart automatically, as of z/OS V1R13.

The sequence of events for internal restart processing is as follows:

1. zFS takes an SAN Volume Controller dump (as it did before, this has not changed).
2. Any new incoming requests to zFS are suspended. Any requests already in the zFS address space are failed.

An application with a failed request might see I/O errors or EAGAIN (“try again later”) until the internal restart is complete.

3. The zFS controller task (IOEFSCM) detaches the zFS kernel subtask (IOEFSKN). At this point, the zFS address space does *not* terminate (this is different from pre-z/OS 1.13). In fact, z/OS UNIX is not even aware that zFS is restarting. Therefore, the mount tree is kept.

zFS file systems that are owned by the restarting system are unowned temporarily. These file systems can be taken over by other systems, depending on your definitions in the BPXPRMxx member.

4. The zFS controller task attaches the zFS kernel subtask.
5. The zFS kernel reinitializes.

The zFS configuration options that are used for reinitialization are the ones that were in effect before the restart. That is, the current zFS configuration options are maintained so they can be used on an internal restart, and IOEFSPRM is not used.

6. The zFS controller task issues the internal mounts. The sequence in which the mounts are issued is based on the number of users being impacted. zFS attempts to get the largest number of users up and running again as quickly as possible. Any zFS file systems that

were previously owned by the restarting system are then taken back after the internal restart is complete. zFS can do dynamic ownership movement, based on the level of zFS activity.

7. Any incoming requests that were suspended are now resumed.

### How to get information about zFS internal restart

In a perfect world, zFS would never have to restart. But there might be times when zFS does an internal restart, and you might not have even noticed. You can gather information about zFS internal restarts by issuing the **MODIFY ZFS,QUERY,STATUS** command. This command can tell you how many internal restarts have occurred since the zFS address space was started. Figure 10-1 shows that there were three internal restarts over the four days since zFS was last started.

```
F ZFS,QUERY,STATUS
IOEZ00736I zFS kernel initiated at Jul 15 16:28:51 2013 308
Current status: active
Internal restart count 3 (Last restart: Jul 19 01:10:09 2013)
IOEZ00025I zFS kernel: MODIFY command - QUERY,STATUS completed
successfully.
```

Figure 10-1 Sample **MODIFY ZFS,QUERY,STATUS** response

### 10.1.3 Restrictions or considerations

If you suspect a problem or hang in zFS, you can manually force an internal restart. Issue the **MODIFY ZFS,ABORT** command only when you suspect a serious internal problem that you want to clear.

As of z/OS 1.13, **MODIFY ZFS,HANGBREAK** will also force an internal restart, meaning that the command behavior has changed. This command has been made equivalent to the **MODIFY ZFS,ABORT** command. Do not use the **MODIFY ZFS,HANGBREAK** command any more. It might cause confusion in that operators might assume that the command will only fail waiting requests that are suspected of being hung (this was the prior behavior). Needless to say, the z/OS 1.13 **MODIFY ZFS,ABORT** command is much more effective at clearing a hang condition than the more limited pre-1.13 behavior of the **MODIFY ZFS,HANGBREAK** command.

### 10.1.4 Rollback

There is no rollback of this function.

### 10.1.5 Further information

For more information about zFS, see *z/OS Distributed File System zSeries File System Administration*, SC23-6887.

## 10.2 zFS automatic re-enable of disabled file systems

This section covers enhancements that allow the automatic re-enablement of disabled file systems in zFS.

## 10.2.1 The issue

Before z/OS 1.13, you might have had to explicitly unmount and then remount a zFS file system that had become disabled. zFS file systems can become disabled for several reasons, such as permanent corruption of the file system, internal zFS error, I/O error on the DASD, or a DASD connectivity problem. You know that a file system has become disabled when you see a message similar to this:

```
IOEZ00422E Aggregate PLEX1.JOE.AGGR001.LDS0001 disabled
```

Disabled file systems affect your system because you cannot read from, or write to, that file system.

If you run an explicit unmount, all file systems in the hierarchy below the unmounted zFS file system must also be unmounted. This has an availability impact because several file systems that were not directly disabled might be involved.

## 10.2.2 How this is addressed in z/OS 1.13

As of z/OS 1.13, zFS decides how to re-enable the file system based on the environment and the condition that is encountered. This is the new behavior of zFS and there is no option to turn it on or off.

### When an internal zFS error is detected or the file system is corrupted

As of z/OS 1.13, when zFS is the owner of disabled file system, zFS attempts to re-enable a disabled file system, so that it can be available for use again. To re-enable the disabled file system, an internal remount using the same mount mode is attempted on the zFS-owning system when one or more of the following are true:

- ▶ The file system environment is not shared.
- ▶ The file system is not sysplex-aware.

**Note:** If you can locally access a shared file system in a specific mount mode, then that file system is said to be “sysplex-aware” in that mount mode.

- ▶ The file system is shared, but no z/OS 1.13 or later systems can take it over.

If none of the above are true, zFS requests that another z/OS 1.11 or later system takes over the file system. In this case, you might see a series of messages similar to this:

```
IOEZ00548I Requesting that SYS2 takeover aggregate PLEX1.JOE.AGGR001.LDS0001
```

You then see messages like these on the system has been requested to take over the file system (SYS2 in this example):

```
IOEZ00388I Aggregate takeover being attempted for aggregate  
PLEX1.JOE.AGGR001.LDS0001
```

```
IOEZ00044I Aggregate PLEX1.JOE.AGGR001.LDS0001 attached successfully.
```

Whether zFS does an internal remount or requests takeover, this behavior has the advantage of allowing zFS to try to maintain availability of the file system, rather than waiting for you to manually respond to it.

### **I/O or connectivity errors**

If an I/O or connectivity error occurs, zFS will not attempt an internal remount with the same mode. zFS relies on I/O recovery or system recovery to handle the problem. You must manually unmount the file system and all the file systems below the disabled file system.

## **10.2.3 Restrictions or considerations**

If zFS detects an error that might damage a file system, it disables the file system to protect it. Monitor message IOEZ00422E because it informs you when a file system becomes disabled. In many cases, this action is successful, and the disabled file system is not damaged.

However, if you detect that a file system has been disabled multiple times, that is an indication of a permanent problem. For these cases, run the salvager utility (**ioeagslv**) to ensure that the zFS file system is internally consistent. The salvager utility requires that the zFS file system is unmounted for this verification.

This raises an interesting dilemma. To optimize availability, zFS now automatically re-enables the file system. However, you must unmount the file system to run salvager after the re-enablement (which makes it unavailable).

This is where your judgement and experience comes in. Weigh the need to have the file system mounted (and possibly internally inconsistent) against having the file system unmounted (and being able to repair any inconsistencies). The longer the file system is accessed and used without the verification of salvager, the more risk you are taking. If the salvager utility is unable to repair the file system, you must recover the file system from a backup that was taken before the damage occurred. Any backups that were taken after the disablement occurred might be internally inconsistent.

## **10.2.4 Rollback**

This function was not rolled back to prior z/OS releases.

## **10.2.5 Further information**

For more information about this capability and for information about the available backup utilities, see *z/OS Distributed File System zSeries File System Administration*, SC23-6887.





## **z/OS enhancements: Security Server (RACF)**

This chapter provides information about enhancements in z/OS Security Server in Resource Access Control Facility (RACF) since z/OS 1.8. These enhancements help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ Sample passphrase ICHPWX11 exit uses System REXX
- ▶ Sample password ICHPWX01 exit uses System REXX

## 11.1 Sample passphrase ICHPWX11 exit uses System REXX

This section covers how the new system component System REXX interacts with the passphrase **ICHPWX11**.

### 11.1.1 The issue

RACF new-password-phrase exit **ICHPWX11** validates passphrase strength against rules that your enterprise requires. When you want to change your passphrase rules, you update this assembly language module, reinstall it, and then run an IPL. Having to wait until the next IPL to enforce passphrase strengths might require you to IPL sooner than you otherwise would have liked.

### 11.1.2 How this is addressed in z/OS 1.9

In z/OS V1R9, a new system component called System REXX was introduced. System REXX provides the ability for REXX execs to run outside of conventional TSO/E and batch environments. System REXX execs are easily updated, and can take effect immediately.

A sample for **ICHPWX11** (in **SYS1.SAMPLIB(RACEXITS)**) can be used to call a System REXX exec **IRRPHREX**. Then, by updating **IRRPHREX**, your changes to the new-password-phrase exit can take place immediately.

### 11.1.3 Restrictions or considerations

There are no known restrictions related to the use of this function.

### 11.1.4 Rollback

No rollback. This support is provided in z/OS V1R9.

### 11.1.5 Further information

See *z/OS Security Server RACF System Programmer's Guide*, SA23-2287 for information about the new-password-phrase exit.

## 11.2 Sample password ICHPWX01 exit uses System REXX

This section covers how the new system component System REXX interacts with the passphrase **ICHPWX01**.

### The issue

RACF new-password exit **ICHPWX01** validates password strength against rules that your enterprise requires. When you want to change your password rules, you update this assembly language module, reinstall it, and then run an IPL. Having to wait until the next IPL to enforce password strengths might require you to IPL sooner than you otherwise would have liked.



## How this is addressed in z/OS 1.9

In z/OS V1R9, a system component called System REXX was introduced. System REXX provides the ability for REXX execs to run outside of conventional TSO/E and batch environments. System REXX execs are easily updated, and can take effect immediately.

A download sample for **ICHPWX01** provides a similar capability that exists for **ICHPWX11** (the new-password *phrase* exit). By using the downloaded sample, you can use **ICHPWX01** to call a System REXX exec **IRRPWREX**. Then, by updating **IRRPWREX**, your changes to the new-password exit can take place immediately.

## Restrictions or considerations

Follow the directions that are provided on the following website for the download sample for **IRRPWREX** and **IRRPWREX**:

<http://www.ibm.com/systems/z/os/zos/features/racf/downloads/rexxpwexit.html>

## Rollback

This capability is available as of z/OS V1R9, with the sample provided as a web download.

## Further information

For more information about RACF exits, see *z/OS Security Server RACF System Programmer's Guide*, SA23-2287.

For more information about the web download sample, see the instructions on the following website:

<http://www.ibm.com/systems/z/os/zos/features/racf/downloads/rexxpwexit.html>





## **z/OS enhancements: ICSF**

This chapter provides information about enhancements in z/OS Integrated Cryptographic Service Facility (ICSF) since z/OS V1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following section:

- ICSF: Improve product stability

## 12.1 ICSF: Improve product stability

The product now cannot be cancelled except by explicit command.

### 12.1.1 The issue

Earlier versions of Integrated Cryptographic Service Facility (ICSF) were cancelable by way of an operator command. If the operator issues the cancel command multiple times, the address space might be terminated before cleanup can be completed. This can cause problems with other ICSF-started tasks that are sharing the key data sets (CKDS, PKDS, TKDS).

Cryptographic Support for z/OS V1R9-R11 was web deliverable.

### 12.1.2 How this is addressed in z/OS 1.12

By making ICSF non-cancelable, operations must issue an explicit command to stop the address space, driving it through normal termination routines.

With HCR7790, a new message, CSFM540I, was added to indicate that a card was fenced off during ICSF initialization.

### 12.1.3 Restrictions or considerations

The ICSF contention is on a **DISPLAY GRS,RES=(qname-rname)** command whenever the ENQ resource specified in the qname-rname option is held, regardless of whether the contention exists. Use it to help determine contention issues:

**DISPLAY GRS,RES=(SYSZCKT.\*)** for CKDS conflicts

**DISPLAY GRS,RES=(SYSZPKT.\*)** for PKDS conflicts

**DISPLAY GRS,RES=(SYSZTKT.\*)** for TKDS conflicts

Health Check performs the following tasks:

- Detects the existence of retained RSA private keys on a PCICC or PCIXCC/CEX2C cryptographic card.

ICSFMIG7731\_ICSF\_RETAINED\_RSAKEY came in z/OS 1.9.

- Verifies that the PKDS size in an ICSF pre-HCR7750 environment is sufficiently allocated to support 4096-bit RSA keys.

ICSFMIG7731\_ICSF\_PKDS\_TO\_4096BIT came in z/OS 1.8.

- Monitors the state of the coprocessors and accelerators daily to detect a negative change in state.

ICSF\_COPROCESSOR\_STATE\_NEGCHANGE came in z/OS 2.1.

- Evaluates the master key states of the coprocessors to detect potential master key problems.

ICSF\_MASTER\_KEY\_CONSISTENCY came in z/OS 2.1.

### 12.1.4 Rollback

This function is not available in previous releases.

### 12.1.5 Further information

For more information about ICSF, see *z/OS Cryptographic Services ICSF: System Programmer's Guide*, SA22-7520.





## z/OS enhancements: JES2

Over the years, there have been constant enhancements to JES2, some of which are intended to let you increase or change the value of most JES2 parameters dynamically.

In general, when new functions are added to JES2, a corresponding change is made to the JES2 commands to let you add or change that function dynamically. Always see the description of the parameter that you want to change in the version of *z/OS JES2 Initialization and Tuning Reference*, SA32-0992, that applies to your installed release of z/OS. Values that required a JES2 restart in the past might support dynamic change now. Also, some require that the multi-access spool (MAS) must be at a certain JES2 level in order for dynamic change to be supported. A few still are not dynamic and require a cold start.

**Important:** You *must* update the JES2 parameter member when you change a value dynamically. JES2 supports dynamically increasing the value of many parameters, but decreasing those values might require a cold start. If you omit to harden a change in the JES2 member, a subsequent MAS-wide restart will appear to JES2 as an attempt to decrease that value, resulting in JES2 thinking that you want to do a cold start.

This chapter includes the following sections:

- ▶ Dynamic JES2 exit support
- ▶ Dynamic changes to JES2 parameters
- ▶ Early release of job log from long running jobs and started tasks
- ▶ Use SPIN=UNALLOC JCL keyword to spin off spool files
- ▶ Ability to stop a JES2 job at the end of the current step
- ▶ Dynamically modify a JES2 spool volume
- ▶ Support for over four billion spin data sets

## 13.1 Dynamic JES2 exit support

The following enhancements have been made to allow dynamic JES2 exit support.

### 13.1.1 The issue

JES2 exits are used for enforcing installation standards or providing functions in support of an ISV product. Before z/OS 1.9, JES2 had to be restarted with a warm or hot Start if you wanted to activate a new or modified JES2 exit, or delete an existing exit.

### 13.1.2 How this is addressed in z/OS 1.9

z/OS 1.9 added support for dynamically adding, replacing, or deleting JES2 exits. This enables a new, simpler process for implementing a new or changed JES2 exit without requiring an outage of the JES2 address space.

For example, assuming that you want to update the function in JES2 exit 5. The code for that function is in a load module called JEXIT1, with an entry point called EXIT05. Define the exit and the load module to JES2 by using the following statements in the JES2 parm member:

```
LOADMOD(JEXIT1)STORAGE=PVT  
EXIT(5) ROUTINE(EXIT05),STATUS=ENABLE,TRACE=YES
```

The JES2 exit name does not have to match the LOAD module name because one load module can potentially support multiple exits.

Assuming (in this example) that you are already using the exit, and want to replace it with a new version, complete the following steps:

1. Assemble the updated exit (with entry point EXIT05) and relink JEXIT1.
2. Refresh LLA.
3. Issue the following JES2 command to refresh the module 'JEXIT1' in the JES2 address space:

```
$T LOADMOD(jexit1),REFRESH
```

4. Then, refresh exit 5. Based on the exit definitions, JES2 knows the entry point name for that exit, and because the load module is loaded in the JES2 address space, it knows where to find that exit routine:

```
$T EXIT(5),REFRESH
```

This enhancement to JES2 also provides other new, related commands:

- ▶ **\$ADD LOADMOD(module)**
- ▶ **\$DEL LOADMOD(module)**

### 13.1.3 Restrictions or considerations

Remember that the new exit module will go into effect at the next IPL or JES2 restart regardless if it is not ready for implementation. Therefore, use a secondary JES2 with a STEPLIB pointing to the library that contains a different named load module with same exit routine for testing before you make any changes to your production JES2.



### 13.1.4 Rollback

APAR OA21346 rolls the Dynamic Exits capability back to z/OS 1.7.

### 13.1.5 Further information

For more information about the commands to support dynamic exits, see *z/OS JES2 Commands*, SA22-7526.

For more information about JES2 exits, see *z/OS JES2 Installation Exits*, SA32-0995.

For more information about the JES2 parameters that are used to define JES2 exits, see *z/OS JES2 Initialization and Tuning Reference*, SA32-0992.

## 13.2 Dynamic changes to JES2 parameters

Enhancements have been made to the JES2 parameters.

### 13.2.1 The issue

Before z/OS 1.2, any started task started with SUB=MSTR could not produce any output to JES2. JES2 creates a “request jobID job” when an address space not running under JES2 requests a JES2 job structure. RACF is an example of a user of this interface. The started task output was not closed until the task was shut down.

### 13.2.2 How this is addressed in z/OS 1.13

z/OS 1.13 also added the SPIN parameter to the REQJobid parameter in JES2. It has same suboptions as documented in section: JOBCLASS(STC) JESLOG SPIN=spinvalue. This can be changed dynamically with \$TREQJobid and displayed with command \$DREQJobid

### 13.2.3 Restrictions or considerations

Specifying SPIN option causes the started tasks to use extra track groups of SPOOL space. Ensure that JES2 has sufficient track groups by issuing the command \$JDDetails to see the high mark of usage.

### 13.2.4 Rollback

This enhancement was not rolled back to earlier releases.

### 13.2.5 Further information

For more information about the commands that support dynamic changing of JES2 parameters, see *z/OS JES2 Commands*, SA22-7526.

For more information about JES2 parameters, see *z/OS JES2 Initialization and Tuning Reference*, SA32-0992.

## 13.3 Early release of job log from long running jobs and started tasks

Enhancements have been made that allow the early release of the job log from long running jobs and started tasks.

### 13.3.1 The issue

Some long running started tasks produce JES-managed spool data sets (JESMSG LG and JESYSMSG) that are later post-processed. In the past, this could cause a number of issues:

- ▶ They might cause the spool to fill because these large spool data sets cannot be deleted until they are closed.
- ▶ The spool data sets cannot be processed until they are closed.
- ▶ The resulting very large files can take a long time to process.

To address these issues, these started tasks were stopped and started purely to close the spool files.

### 13.3.2 How this is addressed in z/OS 1.13

z/OS 1.13 added the ability to spin off JES-managed data sets. This option can be specified on the JOBCLASS JESLOG or JOBREQID JESLOG statements in the JES2 parms member, or on the JOB card.

The spin value can specify a time of day, an interval, or a specified number of lines.

You can also force a JES2 Spin data set by issuing this command for a job:

```
$TJnnnnn,SPIN
```

If DDname= is not specified, all eligible data sets (including JESMSG LG and JESYSMSG) are immediately spun off for the job.

### 13.3.3 Restrictions or considerations

The JOBCLASS SPIN parameter applies only to JES-managed spool data sets. Specifying JESLOG=SPIN causes a job in this class to use at least two extra track groups of SPOOL space for the two JESLOG data sets (JESMSG LG or JESYSMSG). To ensure that there are sufficient track groups, issue the **\$JDDetails** command to display high mark usage.

### 13.3.4 Rollback

This enhancement was not rolled back to earlier releases.

### 13.3.5 Further information

For more information about the commands to spin off spool data sets, see *z/OS JES2 Commands*, SA22-7526.

For more information about JES2 parameters, see *z/OS JES2 Initialization and Tuning Reference*, SA32-0992.

For more information about JES-managed data sets, see *z/OS MVS JCL Reference*, SA23-1385.

## 13.4 Use SPIN=UNALLOC JCL keyword to spin off spool files

Just as it is not possible to process or delete JES-managed spool files until they are closed, spool data sets created by your applications also cannot be processed until they are closed.

### 13.4.1 How this is addressed in z/OS 1.13

To provide the ability to spin off these data sets, z/OS 1.13 added a SPIN=UNALLOC JCL keyword. This keyword allows batch jobs and started tasks to have multiple options to spin SYSOUT data sets. The suboption of UNALLOC identifies the type of SPIN:

SPIN=(UNALLOC,suboption)

UNALLOC has these options:

- ▶ **UNALLOC** with no suboption makes sysout available at unallocation by FREE=CLOSE or at end of STEP.
- ▶ **(UNALLOC,nnn[K|M])** on SYSOUT DD statement is similar to SEGMENT=xxx.
- ▶ **(UNALLOC,DMNDONLY)** indicates that the data set is only to be spun when the operator issues a JES2 spin command: \$TJnnnn with **SPIN** option.
- ▶ **(UNALLOC,time)** indicates the time or interval when the file will be closed

### 13.4.2 How this is addressed in z/OS 1.13

The z/OS 1.13 new SPIN=UNALLOC JCL keyword allows a batch job to have multiple options to spin SYSOUT. The suboptions of UNALLOC identify the type of SPIN:

SPIN=(UNALLOC,suboption)

- ▶ **UNALLOC** with no suboption makes sysout available at unallocation by FREE=CLOSE or at end of STEP.
- ▶ **(UNALLOC,nnn[K|M])** on a SYSOUT DD statement is similar to SEGMENT=xxx.
- ▶ **(UNALLOC,DMNDONLY)** indicates that the data set is only to be spun when the operator issues JES2 spin command: \$TJnnnn with **SPIN** option.
- ▶ **(UNALLOC,time)** indicate at time if 'hh:mm' or every interval if '+hh:mm' (interval greater than 10 minutes)

### 13.4.3 Restrictions or considerations

Specifying SPIN=UNALLOC JCL causes a job in this class to use extra track groups of SPOOL space. Ensure that JES2 has sufficient track groups: issue command \$JDDetails to see high mark of usage.

### 13.4.4 Rollback

This enhancement was not rolled back to earlier releases.

### 13.4.5 Further information

For more information about the commands to support dynamic commands, see *z/OS JES2 Commands*, SA22-7526.

For more information about JCL parameters, see *z/OS MVS JCL Reference*, SA23-1385.

## 13.5 Ability to stop a JES2 job at the end of the current step

This section covers an enhancement that allows you to stop a JES2 job at the end of the current step.

### 13.5.1 The issue

Long running jobs with many job steps that overlap a planned IPL schedule had to be canceled or some process had to ensure that they did not start hours before the IPL schedule. Canceling a job at wrong step can cause excessive restart time.

### 13.5.2 How this is addressed in z/OS 1.13

In z/OS V1.13, the new JES2 restart (**\$E**) command has an option that causes a job to stop at the end of the current step on one image and be requeued for execution on another image:

```
$EJxxxx,STEP[,HOLD]
```

The function uses the new STEP end SSI and can be an inhibit or trigger function with new JES2 exit 58.

### 13.5.3 Restrictions or considerations

This function requires that JES journal be on. Job class must be active in the JES2 Initialization parameters. For JES2 restart command to work, the job must end in the current step.

### 13.5.4 Rollback

This enhancement was not rolled back to earlier releases.

### 13.5.5 Further information

For more information about the commands to support dynamic commands, see *z/OS JES2 Commands*, SA22-7526.

## 13.6 Dynamically modify a JES2 spool volume

You can now dynamically discontinue the use of a JES2 spool volume or increase spool volume size.

### 13.6.1 The issue

Before z/OS 1.13, you had to issue a command to drain the spool volume. It had to wait until all output was either purged through normal processes or manually deleted. With many long-running tasks, a sysout might not be available to purge/delete until the next cycle of the task. If you cancel all work for an inactive volume, JES2 deletes any work remaining on the volume. This process could result in lost spool space on other volumes if the jobs on the volume being canceled have allocated space on other volumes. All other job resources are recovered.

### 13.6.2 How this is addressed in z/OS 1.13

New spool migration moves an existing JES2 spool volume to a new spool volume (move migration) or merges one spool volume into another (merge migration). This enhancement is part of z/OS 1.13 APAR OA36158.

**Attention:** Before attempting a spool migration, you must review the section “\$MSPL - Migrate spool volumes” in *z/OS JES2 Commands*, SA22-7526.

Command to move SPOOL1 volume to new larger SPOOL5 volume:

```
$MSPL(spool1),target=spool5
```

Command to merge SPOOL2,SPOOL3 volumes to existing SPOOL5 volume:

```
$MSPL(spool2,spool3),target=spool5
```

Command to display the status of a spool volume:

```
$DSPL(spool5),status,percent,reserved
```

### 13.6.3 Restrictions or considerations

For restrictions and considerations, review the chapter “\$MSPL—Migrate Spool volumes” in *z/OS JES2 Commands*, SA22-7526.

### 13.6.4 Rollback

This enhancement was not rolled back to releases before z/OS 1.13.

### 13.6.5 Further information

For more information about dynamic commands, see *z/OS JES2 Commands*, SA22-7526.

## 13.7 Support for over four billion spin data sets

Support has been added that can handle over four billion data sets.

### 13.7.1 The issue

With required continuous environments and new SPIN options available, more shops can reach the current JES data set 10M limit per job before the image goes through IPL or long running tasks are recycled. Using the JES2 threshold alerts, manual or automated process have been developed to purge the old output.

**Restriction:** A job non-spin output is not purged until all output is processed.

### 13.7.2 How this is addressed in z/OS 2.1

With z/OS 2.1, the limit was raised over four billion SPIN data sets. To implement, change OUTDEF DSLIMIT to 4B from default 10M:

DSLIMIT=4B

To display the current limit, issued the **\$DOUTDEF** command.

### 13.7.3 Restrictions or considerations

The DSLIMIT setting does not have any effect on the first 10M (9,999,999) SPOOL data sets that are created by a job.

This is implemented with the V2.1 JES2 service level so exit writers can handle the limits.

### 13.7.4 Rollback

The function is rolled back to z/OS 1.10 with APAR OA38944 to support over four billion whereas z/OS 1.13 can also create over four billion. The corresponding SDSF APAR PM59496 rolls back to z/OS 1.10.

### 13.7.5 Further information

For more information about the commands to support dynamic commands, see *z/OS JES2 Commands*, SA22-7526.

For more information about JCL parameters, see *z/OS MVS JCL Reference*, SA23-1385.



## **z/OS enhancements: JES3**

This chapter provides information about enhancements in z/OS JES3 since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ Dynamically modify TCP/IP NJE information
- ▶ Ability to dynamically add spool volumes to JES3
- ▶ Ability to dynamically remove spool volumes from JES3

## 14.1 Dynamically modify TCP/IP NJE information

JES3 network job entry (NJE) allows JES3 users at one location to send jobs to another JES3 location for execution to send output (SYSOUT) data to another JES3 location for processing. NJE also sends jobs, commands, and messages to another JES3 location or to a non-JES3 location.

JES3 provides three types of networking protocols:

- ▶ Binary synchronous communication (BSC) protocols
- ▶ Systems Network Architecture (SNA) protocols
- ▶ As of z/OS 1.8, Transmission Control Protocol/Internet Protocol (TCP/IP) protocols

### 14.1.1 The issue

With only two types of networking protocols, BSC and SNA, JES3 NJE was not able to participate in the industry standard communication protocol of TCP/IP that many clients rely upon for network traffic and are familiar with supporting. Supporting multiple and different types of networks within an enterprise was more work and more difficult for clients, and was an important JES3 requirement from many clients. In addition, BSC and SNA hardware were reaching the end of their service period.

### 14.1.2 How this is addressed in z/OS 1.8

With the addition of TCP/IP protocols in JES3, NJE has expanded its reach into client enterprises, allowing more industry standard communication methods. JES3 TCP/IP NJE takes advantage of all the availability enhancements and advances that TCP/IP offers. Also, clients can remove their BSC and SNA hardware after they begin using TCP/IP.

As client networks expand and change, JES3 TCP/IP NJE can dynamically change too. At a high level, use the following steps to implement JES3 TCP/IP NJE:

1. Define a network server, using the **NETSERV** statement.
2. Define a node that specifies it will use TCP/IP communication protocol, using the **NJERMT** statement.
3. Define a socket on one node to talk to another node, by using the **SOCKET** statement.
4. Start the Netserv address space by using TCP Dynamic Support Program.
5. Activate the socket to begin communication between nodes.

The **NETSERV**, **SOCKET**, and **NJERMT** node definitions can be changed dynamically to accommodate changes that occur in the client network:

- ▶ For network server information:
  - Use **\*MODIFY,NETSERV,ADD=** to dynamically add a TCP/IP/NJE Network Server.
  - Use **\*MODIFY,NETSERV=** to modify an existing network server definition. An active network server cannot be modified except to change the **JTRACE**, **VTRACE**, or **ITRACE** parameter.
  - Use **\*MODIFY,NETSERV,DELETE=** to dynamically delete a TCP/IP/NJE Network Server. An active network server cannot be deleted except to change the **JTRACE**, **VTRACE**, or **ITRACE** parameter.



- ▶ For socket information:
  - Use **\*MODIFY,SOCKET,ADD=** to dynamically add a TCP/IP/NJE Network Server.
  - Use **\*MODIFY,SOCKET=sockname** to modify an existing socket definition. An active socket cannot be modified except to change the JTRACE, VTRACE, or ITRACE parameter.
  - Use **\*MODIFY,SOCKET,DELETE=** command to dynamically delete a TCP/IP/NJE socket. An active socket cannot be deleted except to change the JTRACE, VTRACE, or ITRACE parameter.
- ▶ For node information, use **\*MODIFY,NJE,keyword** where *keyword* is:
  - **ADD=** to add information about a node.
  - **DEL=** to delete information about a node,
  - Many other values

Using this suite of commands allows JES3 NJE to accommodate changes in the client Internet Protocol network without a need for a JES3 restart or IPL.

### 14.1.3 Restrictions or considerations

There are no known restrictions related to the use of this function.

### 14.1.4 Rollback

z/OS 1.8 provides TCP/IP NJE. This function is not rolled back.

### 14.1.5 Further information

See *z/OS JES3 Commands*, SA22-7540, for more information about the **\*MODIFY** command.

See *z/OS JES3 Initialization and Tuning Reference*, SA22-7552, for more information about the NETSERV, SOCKET, and NJERMT statements.

## 14.2 Ability to dynamically add spool volumes to JES3

To maintain system availability, having enough spool volumes for JES3 to read jobs from and write job output to is critical. Being able to add spool volume extents dynamically to JES3 plays a large role in allowing JES3 to support a longer time between z/OS IPLs.

**Note:** You *add* spool volumes or extents to the JESPLEX. You are not actually *allocating* spool volume or extent data sets themselves. Do the allocation before the spool volume is added, as indicated in the usage steps below.

### 14.2.1 The issue

Before z/OS 1.13, adding a JES3 spool extent (volume) or removing one from the JESplex was disruptive. To add a spool extent requires a z/OS IPL and a warm start of the JES3 global main, which is disruptive to all of the systems in the JESplex. Then, an IPL on the other

systems in the JESplex was necessary, as well as a restart of all those JES3 local mains. In other words, a complex-wide IPL was necessary when adding a spool.

## 14.2.2 How this is addressed in z/OS 1.13

z/OS 1.13 JES3 provides the capability to add a spool extent with the following steps:

1. Create and format a spool extent, as usual. There is no change here.
2. Create or update the JES3 spool access method (JSAM) member and initialization (inish) deck to indicate information about the new spool extent.
3. Activate the new spool extent in one of these ways:
  - Without an IPL with the **\*MODIFY CONFIG** command
  - With a hot start with refresh

Using either of these methods does not require an IPL for the z/OS 1.13 JES3 mains.

After all of the restarted JES3 locals connect or have been restarted, all added spool extents and partitions are available and used.

### Creating the JSAM

Creating a JSAM initialization stream member allows you to use the same JSAM member for subsequent JES3 starts. In addition, by having the JSAM member hardened into a data set, you are able to use the same JSAM member for subsequent JES3 starts that read the inish deck.

Your new JSAM member should have these characteristics:

- ▶ Start with DYNALLOC statements
- ▶ End with ENDJSAM

All of your DYNALLOC statements should be contained in the same JSAM member, without any “nesting” of other DYNALLOC statements. That is, do not have an INCLUDE statement that contains another INCLUDE statement.

An example of a simple JSAM member is shown in Example 14-1.

*Example 14-1 JSAM member statements to add a spool extent*

---

```
DYNALLOC,DDN=IATPLBI1,DSN=SYS1.PROCLIB
...
TRACK,DDNAME=SPOOLX,SPART=PARTX
...
SPART,NAME=PARTX,DEF=YES,INIT=YES,JOBN0(1,32767,32767),MT=ON
...
ENDJSAM
```

---

Your inish deck will have an INCLUDE statement, as shown in Example 14-2.

*Example 14-2 Inish example for the JSAM member*

---

```
...
INCLUDE,MEMBER=JSAM
...
```

---

This JSAM member and inish deck cannot be used on pre-z/OS 1.13 systems.

### Using the MODIFY command to add spool extents

After you set up the JSAM and inish deck, you can add the spool with the **MODIFY** command:

```
*MODIFY CONFIG,ADD=JSAM
```

When JES3 processes this request, it evaluates what you specified in your JSAM member: New extents are added, new partitions are added, and any OPTIONS parameters are changed. If no errors are found, the JES3 commits the changes.

New spool extents are available after each local is connected or flushed. A flush happens automatically when XCF detects that the system has left the sysplex.

### Using a hot start with refresh to add spool extents

If you prefer to do a hot start with refresh, make sure the inish deck used includes the JSAM updates for the addition of the new spool extent. JES3 local restarts are not necessary for adding a spool because they are automatically restarted by the JES3 global.

Any errors that are encountered during a hot start with refresh require restarting JES with a hot start to restore it to the original configuration. After that, you can fix the problem and attempt to add the spool extents again.

## 14.2.3 Restrictions or considerations

To help avoid errors during JES3 initialization, JES3 provides a utility called the initialization stream checker. This utility simulates the JES3 initialization process and enables you to verify your initialization stream before initializing JES3. Use the JES3 initialization stream checker utility to test changes made to your JES3 inish deck.

## 14.2.4 Rollback

The capability to add spool extents to JES3 is provided in z/OS 1.13, and is not rolled back. The associated capability to remove spool extents from JESplex usage is introduced in z/OS 2.1. For more information, see 14.3, “Ability to dynamically remove spool volumes from JES3” on page 161.

## 14.2.5 Further information

See *z/OS JES3 Commands*, SA22-7540 and *z/OS JES3 Initialization and Tuning Reference*, SA22-7552, for more detailed information about dynamic spool addition.

## 14.3 Ability to dynamically remove spool volumes from JES3

Adding spool volumes (extents) to JES3 is important, and having the ability to remove spool volumes from the JESplex is important too. Both the addition and the removal capability allows system programmers to dynamically maintain spool extents according to the needs of the system, lengthening the time between IPLs.

**Remember:** You remove spool volumes or extents from the JESPLEX. You are not actually deleting spool volume or extent data sets themselves.

### 14.3.1 The issue

Just as adding JES3 spool extents was disruptive until z/OS 1.13, removing spool extents is similarly disruptive to the JESplex. Removing extents requires an IPL and a warm start of the JES3 global main, followed by an IPL and restart of all the JES3 local mains.

### 14.3.2 How this is addressed in z/OS 2.1

Like the add spool extent support in z/OS 1.13 JES3, removing spool extents from the JESplex can be done dynamically by using the **\*MODIFY CONFIG** command or using hot start with refresh. Spool extents can be added and removed at the same time. Use the following steps to perform the dynamic removal:

1. Drain the spool extent to be remove.
2. Remove job data from the spool extent.
3. Create or update the JSAM member and initialization (inish) deck to indicate the spool extent you want to remove.
4. Remove the spool extent in one of two ways:
  - Without an IPL with the **\*MODIFY CONFIG** command
  - With a hot start with refresh

Using either of these methods does not require an IPL for the JES3 mains. JES3 locals do *not* have to reconnect or be restarted if they are z/OS R13 or higher.

#### Draining the spool extent

Before removal, use the **DRAIN** command on the spool extent. The **DRAIN** command stops JES3 from allocating tracks in the spool extent, and moves existing single track table (STT) records to another spool extent within the default partition. A sample command is:

```
*MODIFY Q,DD=ddname,DRAIN
```

#### Removing job data from the spool extent

To remove job data from the spool extent, first determine which jobs have data in the extent. A sample command to do that is:

```
*INQUIRY Q,DD=ddname,U,N=nnn
```

Then, dump or purge all job data from the extent. A sample command to dump the job data might be:

```
*START,DJ,DD=ddname
```

#### Creating the JSAM

As mentioned previously (in 14.2, “Ability to dynamically add spool volumes to JES3” on page 159), creating a JSAM initialization stream member allows you to use the same JSAM member for subsequent JES3 starts. For dynamic spool extent removal, your new JSAM member should have these characteristics:

- ▶ Start with DYNALLOC statements
- ▶ End with ENDJSAM

Place *all* of your DYNALLOC statements in the same JSAM member when doing a removal. This is different from doing a dynamic spool addition, where JSAM members might have only

DYNALLOC statements. For removals, non-spool data sets must still be included. JES3 will fail the dynamic spool change if any statements are changed or missing.

As with dynamic addition, do not “nest” other DYNALLOC statements by having an INCLUDE inside another INCLUDE statement.

An example of a simple JSAM member is shown in Example 14-3.

*Example 14-3 JSAM member statements to remove a spool extent*

---

```
DYNALLOC,DDN=IATPLBI1,DSN=SYS1.PROCLIB
...
TRACK,DDNAME=SPOOLY,SPART=PARTY
...
SPART,NAME=PARTY,DEF=YES,INIT=YES
...
OPTIONS,WANTDUMP=YES,JOBNO=(65500,999999,10000),XCFGRPNUM=JESXCFX3,
DUMP=PRDMP,MT=ON
...
BUFFER,BUFSIZE=4084,GRPSZ=20,PAGES=(1024,64,256),MINBUF=32,SPLIM=(10,20)
...
ENDJSAM
```

---

Then, your inish deck will have an INCLUDE statement as shown in Example 14-4.

*Example 14-4 Inish example for the JSAM member*

---

```
...
INCLUDE,MEMBER=JSAM
...
```

---

Again, similar to a dynamic spool addition, this JSAM member and inish deck cannot be used on pre-z/OS 1.13 systems.

## Using the MODIFY command to remove a spool extent

After you set up the JSAM and inish deck, you can remove the spool with the **MODIFY** command:

```
*MODIFY CONFIG,M=JSAM
```

**M=** has been added for z/OS 2.1. You can use it to add or remove a spool extent. You can continue to use **ADD=JSAM**, if you want.

When JES3 processes this request, it evaluates all parameters that qualify for dynamic modification. Then, spool extents to be removed are identified. New spool extents might also be identified. You must confirm the removal of spool extent by responding to WTOR message:

```
IAT4000 CONFIRM DELETION OF SPOOL DATASET ddnn (CONTINUE OR CANCEL)
```

JES3 verifies that no job or dynamic support program (DSP) data exists in the extent to be removed. If job data is found, it will not be moved. You must then purge or drain the job or data before trying again. When no errors have occurred during verification and you have confirmed the removal, the extent is removed from the JESplex.

z/OS 1.13 (or later) JES3 locals are automatically restarted by the JES3 global without operator intervention. No IPL is required. However, if you have a pre-z/OS 1.13 JES3 local in the JESplex, then you must IPL and restart each of them.

### Using a hot start with refresh to remove a spool extent

If you prefer to do a hot start with refresh, make sure the inish deck used includes the JSAM updates for the removal of the wanted spool extent. z/OS 1.13 (or later) JES3 locals are automatically restarted by the JES3 global, without operator intervention. No IPL is required. If you have a pre-z/OS 1.13 JES3 local in the JESplex, you must IPL and restart them.

Any errors that are encountered during a hot start with refresh require restarting JES with a hot start to restore to the original configuration. After that, you can fix the problem and attempt to remove the spool extent again.

### 14.3.3 Restrictions or considerations

As mentioned previously, place *all* of your DYNALLOC statements in the same JSAM member when doing a removal. If any dynamic spool change statements are missing, the dynamic spool change fails.

Use the JES3 initialization stream checker utility to test changes made to your JES3 inish deck.

The command **\*F CONFIG,M=member,LOG=YES** can be used to check the contents of the JSAM member and create a log data set. The command can be ended before any spool changes are actually made.

### 14.3.4 Rollback

The support to dynamically remove spool extents from the JESplex was introduced in z/OS 2.1 JES3. However, support is provided for z/OS V1R13 JES3 mains and locals to accommodate dynamic remove spool extents.

### 14.3.5 Further information

For more detailed information about dynamic spool removal, see *z/OS JES3 Initialization and Tuning Reference*, SA22-7552, and *z/OS JES3 Commands*, SA22-7540.



## **z/OS enhancements: Infoprint Server**

This chapter provides information about enhancements in z/OS Infoprint Server since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ Dynamic configuration changes for most InfoPrint options
- ▶ Print Services Facility (PSF) installation is disassociated from SYS1.NUCLEUS

## 15.1 Dynamic configuration changes for most InfoPrint options

Infoprint Server allows you to print files on z/OS printers from any workstation that has TCP/IP access. This z/OS priced exclusive feature consists of the following components or functions:

- ▶ IP IBM PrintWay™ basic mode
- ▶ NetSpool
- ▶ Print Interface
- ▶ Printer Inventory Manager
- ▶ Transform Interface
- ▶ z/OS InfoPrint Central

### 15.1.1 The issue

When changes are made to configuration attributes, Infoprint Server must be stopped and restarted for the configuration changes to become active. Infoprint Server needed to provide a mechanism to allow changing of configuration attributes without requiring a restart of Infoprint Server.

### 15.1.2 How this is addressed in z/OS 2.1

Infoprint Server provides a function to allow dynamic changes to several configuration options without requiring a restart of the Infoprint Server daemons.

After the dynamic configuration function is enabled for use, authorized administrators can easily update dynamic attributes with the Interactive System Productivity Facility (ISPF) panels or the Printer Inventory Definitions Utility (PIDU). The configuration attribute changes can take effect in most cases without requiring a restart of Infoprint Server or its daemons.

#### Enabling the dynamic configuration function

Before turning on the dynamic configuration function, ensure that upgrade to release z/OS V2R1 is complete and fallback to an older release is not likely.

Dynamic configuration is enabled by completing these steps:

1. Edit `aopd.conf` file and add `'dynamic-configuration = yes'`. (`'dynamic-configuration = no'` is the default.) Save the `aopd.conf` file.
2. If Infoprint Server is running, stop all daemons by using the **AOPSTOP** command.
3. Use **AOPSTART** to start Infoprint Server. When Infoprint Server starts, it automatically creates the system configuration definitions in the Printer Inventory.

If you want to turn off the dynamic configuration function (`'dynamic-configuration=no'`) for some reason, follow the steps in *z/OS Infoprint Server Customization*, S544-5744.

#### Changing dynamic attributes

Dynamic attributes are configuration attributes that are stored in the system configuration definition in the Printer Inventory. When dynamic configuration is enabled, you can change the dynamic attributes while Infoprint Server is running.

If you change a dynamic attribute, with a few exceptions, the new value takes effect when you save the change. A few attributes require that you stop and restart the daemons that use the attribute before the change can take effect. For information about those exceptions, see *z/OS*



*Infoprint Server Customization*, S544-5744. At the time of writing, of the 36 dynamic attributes, 22 of them needed no daemon restarts whatsoever, and 14 needed one or more daemons to be restarted.

To edit the dynamic attributes, complete these steps:

1. Use one of these methods to edit the system configuration attributes:
  - a. Use Infoprint Server ISPF panel or PIDU
  - b. Use ISPF to navigate to the Infoprint Server ISPF panel. Then, select option 8 – “Manage System Configuration”.
2. Infoprint Server startup information can be viewed and changes can be made. After the changes are complete, save the changes before exiting.
3. If the change requires a restart to an Infoprint Server daemon, stop the daemon and restart that daemon for the change to take effect. See the *z/OS Infoprint Server Customization*, S544-5744, for more information about which daemons require a restart for which dynamic attributes.

The `aopd.conf` configuration file is still needed when dynamic configuration is enabled. Some configuration parameters are still required to remain in this file.

### 15.1.3 Restrictions or considerations

Dynamic changes are *only* allowed when running in IP PrintWay extended mode. That is, you *cannot* perform dynamic changes in IP PrintWay basic mode.

To determine whether the dynamic configuration capability is enabled, use these methods:

- ▶ If Infoprint Server is running, use its ISPF panel option 12.8 to display Infoprint Server: Inventory Manager, and look for option 8 (Manage system configuration) at the bottom of the display. If dynamic configuration is not enabled, option 8 will not be displayed.
- ▶ Use PIDU to view the system configuration definitions in the Printer Inventory.

### 15.1.4 Rollback

This enhancement is provided in z/OS V2R1 and is not rolled back.

### 15.1.5 Further information

For more information about enabling the dynamic configuration option and using it, see *z/OS Infoprint Server Customization*, S544-5744.

## 15.2 Print Services Facility (PSF) installation is disassociated from SYS1.NUCLEUS

Print Service Facility (PSF) is a licensed program that runs on z/OS. The current release at time of writing is PSF V4.4 (5655-M32). PSF contains some load modules that install into the SYS1.NUCLEUS data set.

### **15.2.1 The issue**

Because PSF has nucleus load modules, it is unable to be separated into its own data sets and shares the SYS1.NUCLEUS data set with z/OS. Due to its inclusion in SYS1.NUCLEUS, it is “tied” to a z/OS level, which makes installation and activation of a different PSF level impossible without an IPL.

### **15.2.2 How this is addressed in z/OS V2.1**

As of z/OS V2R1 with APAR OA42255, z/OS Infoprint Server (an exclusive priced feature of z/OS) supplies the PSF nucleus load modules.

With the load modules that PSF needs now owned by z/OS and loaded into the nucleus by z/OS itself, PSF has been disassociated from the SYS1.NUCLEUS data set. PSF now installs into unique data sets and can be installed and activated without a z/OS IPL.

### **15.2.3 Restrictions or considerations**

This coordinated packaging improvement between z/OS and PSF allows PSF to be handled similar to other licensed programs that have separate data sets from z/OS. PSF upgrades might not require a z/OS IPL. See the PSF documentation for installation and upgrade considerations.

### **15.2.4 Rollback**

There is no rollback of this support. z/OS V2R1 with the PTF for OA42255 on Infoprint Server provides this support for PSF.

### **15.2.5 Further information**

For more information about the installation and activation of PSF, see the PSF licensed program publications, including the program directory.



## **z/OS enhancements: TSO/E**

This chapter provides information about enhancements in z/OS Time Sharing Option Extensions (TSO/E) since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following section:

- Support for VTAM unconditional reconnect for TSO/E users

## 16.1 Support for VTAM unconditional reconnect for TSO/E users

Recent enhancements have added support for VTAM unconditional reconnect for TSO/E users.

### 16.1.1 The issue

TSO/E users would sometimes find themselves “hung” on their VTAM connection. The user would try to log on to TSO/E with **RECONNECT** but would often receive this error message:

```
IKJ56411I TSOLOGON RECONNECT REJECTED - USERID IBMUSER IN USE
```

This is frustrating and often results in a call to operations to cancel the user ID. This certainly would not result in a system outage (or cause an IPL), but does result in a user ID not being available. It can therefore be viewed as a “user ID outage”.

The issue is that VTAM would not detect that there was a disconnect, and therefore not allow you to reconnect. One of the reasons **RECONNECT** fails is that the system cannot detect the user is in a disconnected state, or at least not in a timely manner. This might happen if the terminal emulator crashes, or if someone tries to log on from another computer without disconnecting first, or their IP address is renewed.

### 16.1.2 How this is addressed in z/OS 1.11

As of z/OS V1R11, TSO/E **LOGONHERE** support for VTAM unconditional reconnect now allows you to reconnect to your session even if no disconnection has been detected and you have not canceled your previous session. By default in z/OS V1R11, **LOGONHERE** support is turned on. By specifying the reconnect option, you can easily switch from one computer to another or reestablish a session after a loss of connectivity (even with a new IP address).

To control this, a new PARMLIB option, **LOGONHERE (ON|OFF)**, has been added under the **LOGON** statement in **IKJTS0xx**. The default is ON, which should reduce the number of times that operators must cancel TSO/E user IDs. However, the old behavior can be restored by setting the value to OFF.

### 16.1.3 Restrictions or considerations

Verify your **TSOKEYxx RECONLIM=** setting to make sure that it is nonzero. **RECONLIM** specifies the time limit in minutes within which a user can reconnect after the TP line is disconnected. The default setting of **RECONLIM=0** means that there is a zero wait for reconnection, which means that a reconnect is not possible. The value of **RECONLIM** is 0-32767, and the default is 3.

To see your current setting, issue **D IKJTS0,PARMLIB**, as seen in Example 16-1.

*Example 16-1 D IKJTS0,LOGON example response*

---

**D IKJTS0,LOGON**

IKJ738I TSO/E PARMLIB SETTINGS : 176

SYS1.PARMLIB(IKJTS000) on volume #@\$#M1

Activated by \*\*IPL\*\* on 2013-08-01 at 14:57:30 from system #@\$A

Applies to : #@\$A

CURRENT PARMLIB SETTINGS FOR LOGON:

PASSPHRASE(ON)

VERIFYAPPL(OFF)  
LOGONHERE(ON)

---

You can dynamically change the setting by using **PARMLIB UPDATE** or **SET IKJTSO**.

#### **16.1.4 Rollback**

This support is provided in z/OS V1R11 and later.

#### **16.1.5 Further information**

For more information about this enhancement, see *z/OS TSO/E Customization*, SA22-7782.





## **z/OS enhancements: z/OS UNIX**

This chapter provides information about enhancements in z/OS UNIX since z/OS 1.8 that help you avoid or postpone a planned outage of z/OS or one of the components of z/OS.

This chapter includes the following sections:

- ▶ Support dynamic modification of AUTOCDT setting
- ▶ Prevent content overlay during mount
- ▶ Change sysplex root data set without sysplex IPL
- ▶ Support for alternate sysplex root data set
- ▶ Change number of common inet ports without OMVS restart
- ▶ Remount with same mount mode

## 17.1 Support dynamic modification of AUTOCVT setting

Enhancements have been made that support dynamic modification of the AUTOCVT setting.

### 17.1.1 The issue

z/OS is an EBCDIC environment with EBCDIC devices and does not naturally use ASCII. You can use the **iconv** command to change the data from EBCDIC to ASCII and back, or code *AUTOCVT(ON)* in BPXPRMxx, which only supports EBCDIC 1047 and ASCII 819 code pages.

Users require access to Enhanced ASCII or Unicode Services so they can run EBCDIC to ASCII conversions within their application. If you want to change your systems' defaults, you have no idea if there are files that users have that are tagged for ASCII conversion. You also do not know the affect of the change will be on system, task, or user performance.

### 17.1.2 How this is addressed in z/OS V1.13

This is either *AUTOCVT(OFF)*, *AUTOCVT(ON)*, or *AUTOCVT(ALL)*.

### 17.1.3 Restrictions or considerations

Use *AUTOCVT* to enable Enhanced ASCII or Unicode Services. Although the default in BPXPRMxx is still OFF, you can code **ALL** to enable Unicode Services support. Every read or write operation in UNIX is then checked to see whether a conversion is needed. This can cause a performance issue if files are marked for conversion or if an entire file system is tagged.

You can use the **setomvs autocvt=off|on|all** command to alter the system definition. Using the **setomvs omvs=(xx)**, you can load a new BPXPRMxx member.

Alternately, `_BPXK_AUTOCVT` was enhanced to support the change to the system option AUTOCVT. A user can now set this symbol in the UNIX .profile if they need to override the system entry.

In most cases, limit this function to the smallest set of users possible. Setting AUTOCVT(OFF) in BPXPRMxx allows the user (or task) to control it themselves. Because your UNIX environment can support hundreds of users, this seems the most prudent direction.

### 17.1.4 Further information

For more information about the AUTOCVT setting, see *z/OS MVS Initialization and Tuning Reference*, SA23-1380.

You can find information about the practical use of AUTOCVT and `_BPXK_AUTOCVT` in *z/OS V2 R1 UNIX SYSTEM SERVICES*, GA32-0884.

## 17.2 Prevent content overlay during mount

This section covers enhancements that help prevent content overlay during mount.



## 17.2.1 The issue

z/OS UNIX file systems can mount over a directory that already contains files and directories. When a mount is done on a non-empty mount point, it is possible to “cover up” the existing files and directories with the contents of the newly mounted file system.

The exposure to be concerned about is data availability. After the file and directories in a non-empty mount point are inaccessible, applications might (or might not) have access to that data.

## 17.2.2 How this is addressed in z/OS 1.13

As of z/OS V1R13, z/OS UNIX can notify you when you are mounting a file system on a non-empty mount point. You can use this information to be informed of possible data availability problems, and take appropriate actions.

The BPXPRMxx PARMLIB statement **NONEMPTYMOUNTPT** can be used to control how the system mounts the file systems on non-empty mount points.

- ▶ The **NOWARN** option specifies that the mount is to take place without any warning message when the mount point is a non-empty directory. The contents of that directory are hidden during the mount. This is the default setting.
- ▶ The **WARN** option specifies that the mount is to take place with a warning message when the mount point is a non-empty directory. The contents of that directory are hidden during the mount.
- ▶ The **DENY** option specifies that mounting is not to take place when the mount point is a non-empty directory.

During OMVS initialization, if the mount point is contained in an NFS file system, the **NONEMPTYMOUNTPT** setting is not honored.

**D OMVS,OPTIONS** shows the current option for **NONEMPTYMOUNTPT** in use, as seen in Example 17-1.

*Example 17-1 D OMVS,OPTIONS sample response*

---

```
D OMVS,OPTIONS
BPX0043I 16.22.52 DISPLAY OMVS 076
OMVS      0010 ACTIVE              OMVS=(00,F1)
CURRENT UNIX CONFIGURATION SETTINGS:
MAXPROCSYS      =      1000      MAXPROCUSER      =      50
MAXFILEPROC     =      64000     MAXFILESIZE      = NOLIMIT
MAXCPUPTIME     = 2147483647     MAXUIDS       =      50
MAXPTYS         =      256      MAXIOBUFUSER      =      2048
MAXMMAPAREA     =      40960     MAXASSIZE      = 2147483647
MAXTHREADS      =     100000     MAXTHREADTASKS =      32767
...
AUTHPGMLIST     = NONE
SWA              = BELOW        NONEMPTYMOUNTPT = NOWARN
SERV_LINKLIB    =
SERV_LPALIB     =
...
```

---

## Using the WARN option

**WARN** can be set dynamically by using the **SETOMVS** or **SET OMVS** commands.

Example 17-2 shows setting **WARN** dynamically and then verifying the option with **D OMVS,OPTIONS**.

*Example 17-2 Example scenario of setting WARN and verifying the result*

---

**SETOMVS NONEMPTYMOUNTPT=WARN**

BPX0015I THE SETOMVS COMMAND WAS SUCCESSFUL.

**D OMVS,OPTIONS**

BPX0043I 17.06.35 DISPLAY OMVS 203

OMVS 0010 ACTIVE OMVS=(00,F1)

CURRENT UNIX CONFIGURATION SETTINGS:

MAXPROCSYS	=	1000	MAXPROCUSER	=	50
------------	---	------	-------------	---	----

MAXFILEPROC	=	64000	MAXFILESIZE	=	NOLIMIT
-------------	---	-------	-------------	---	---------

MAXCPUPTIME	=	2147483647	MAXUIDS	=	50
-------------	---	------------	---------	---	----

MAXPTYS	=	256	MAXIOBUFUSER	=	2048
---------	---	-----	--------------	---	------

MAXMMAPAREA	=	40960	MAXASSIZE	=	2147483647
-------------	---	-------	-----------	---	------------

...

AUTHPGMLIST	=	NONE
-------------	---	------

SWA	=	BELOW
-----	---	-------

**NONEMPTYMOUNTPT = WARN**

SERV_LINKLIB	=
--------------	---

SERV_LPALIB	=
-------------	---

...

---

When **WARN** is in effect, a mount at a non-empty mount point succeeds and the following message is found in the system log:

BPXF263I file system 896

IBM.DATA.FILE.SYS

HAS BEEN MOUNTED ON A NONEMPTY DIRECTORY

## Using the DENY option

**DENY** can be set dynamically by using the **SETOMVS** or **SET OMVS** commands.

Example 17-3 is an example scenario of setting **DENY** dynamically, and then verifying the option with **D OMVS,OPTIONS**.

*Example 17-3 Example scenario of setting DENY and verifying the result*

---

**SETOMVS NONEMPTYMOUNTPT=DENY**

BPX0015I THE SETOMVS COMMAND WAS SUCCESSFUL.

**D OMVS,OPTIONS**

BPX0043I 17.06.35 DISPLAY OMVS 203

OMVS 0010 ACTIVE OMVS=(00,F1)

CURRENT UNIX CONFIGURATION SETTINGS:

MAXPROCSYS	=	1000	MAXPROCUSER	=	50
------------	---	------	-------------	---	----

MAXFILEPROC	=	64000	MAXFILESIZE	=	NOLIMIT
-------------	---	-------	-------------	---	---------

MAXCPUPTIME	=	2147483647	MAXUIDS	=	50
-------------	---	------------	---------	---	----

MAXPTYS	=	256	MAXIOBUFUSER	=	2048
---------	---	-----	--------------	---	------

MAXMMAPAREA	=	40960	MAXASSIZE	=	2147483647
-------------	---	-------	-----------	---	------------

...

AUTHPGMLIST	=	NONE
-------------	---	------

SWA	=	BELOW
-----	---	-------

**NONEMPTYMOUNTPT = DENY**

```
SERV_LINKLIB    =  
SERV_LPALIB     =  
...
```

---

When **DENY** is in effect, a mount at a non-empty mount point fails, providing this message to the TSO/E user who requested the mount:

```
BPXF135E RETURN CODE 00000088, REASON CODE 055B063C. THE MOUNT FAILED FOR file  
system IBM.DATA.FILE.SYS.
```

Using **bpxmtext 063C** provides the user with the reason for the mount failure as shown in Example 17-4.

*Example 17-4 bpxmtext 063C response*

---

```
Notice: unknown modid, reason text might be incorrect  
JrNonEmptyMntPtDir: The mount point directory is not empty.  
Action: Retry the mount on an empty mount point directory.
```

---

### 17.2.3 Restrictions or considerations

Notice the difference in message response locations when using **WARN** and **DENY** are used. **WARN** messages are found in the system log and are not provided directly back to the TSO/E user who ran the mount. **DENY** messages are provided directly back to the user and are not provided in the system log.

If you use the Nonprivileged User Mount function (introduced in z/OS V1R13), those mounts *must* be at an empty mount point. That is, a nonprivileged user can never mount on a non-empty mount point.

### 17.2.4 Rollback

This function is available in z/OS 1.13 and has not been rolled back.

### 17.2.5 Further information

For more information about using this enhancement, see *UNIX System Services Planning*, GA32-0884.

For more information about the **SETOMVS** and **SET OMVS** commands, see *z/OS MVS System Commands*, SA38-0666.

## 17.3 Change sysplex root data set without sysplex IPL

You can now change the sysplex root data set without a sysplex IPL.

### 17.3.1 The issue

To implement a new root file system, for example to have it on a different volume, make it larger, or to change the format, in previous releases, you had to shut down the sysplex.

### 17.3.2 How this is addressed in z/OS 1.10

Using a **F OMVS,NEWROOT=new.root.file.system.name,COND=<Yes|No>** command, you can swap the SYSPLEX ROOT file system.

Update the TYPE parameter and name of the sysplex root file system in the BPXPRMxx member of SYS1.PARMLIB.

**Important:** Old connections in the previous sysplex root file system might get EIO errors.

To display information about the sysplex root, issue this command:

```
D OMVS,F,N=fsroot_data_set_name
```

### 17.3.3 Restrictions or considerations

The new SYSPLEX ROOT needs to have all the same directories and symbolics as the one in use. You need to ensure that all systems in your sysplex have access to it.

### 17.3.4 Rollback

This has not been rolled back to previous releases.

### 17.3.5 Further information

For a description of the new sysplex root, see *UNIX System Services Planning*, GA32-0884.

For information about the command, see *z/OS MVS System Commands*, SA38-0666.

## 17.4 Support for alternate sysplex root data set

Support has been added for an alternate sysplex root data set.

### 17.4.1 The issue

Because of the many system services that depend on UNIX, any outage can be dramatically disruptive to the environment. Having an alternate SYSPLEX ROOT file system available minimizes potential disruptions that might result.

### 17.4.2 How this is addressed in z/OS 1.10

This allows you to define an alternate for your sysplex root file system. Although it does not have to be in the same format (HFS versus zFS), it needs to have all of the currently used mount points and aliases in place that, if it is used, nothing is lost. To eliminate a DASD issue, you need to ensure that the alternate sysplex root is not in the same volume, device, and control unit as the current sysplex root.

You can define this in BPXPRMxx and you can use the **SET OMVS=(xx)** to enable it. You might want to use the **SETOMVS SYNTAXCHECK=(xx)** command to validate the ALTROOT syntax.

To display information about the sysplex altroot, issue this command:

```
D OMVS,F,N=fsroot_alt_data_set_name
```

### 17.4.3 Restrictions or considerations

All systems in the sysplex need to have access to the alternate root. z/OS UNIX will not allocate the alternate root if it is not available at start or when you enter the command. Although it does not need to be in the same data set format as the current SYSPLEX ROOT, it does need to have all current mount points and symbolics defined before use.

**Tip:** The alternate sysplex root needs to remain current, so build a process that provides updates to the sysplex root and the alternate sysplex root at the same time.

### 17.4.4 Rollback

This has not been rolled back to prior releases.

### 17.4.5 Further information

You can find a description of the alternate sysplex root in *UNIX System Services Planning*, GA32-0884.

## 17.5 Change number of common inet ports without OMVS restart

Recent enhancements allow you to change both MAXSOCKETS and INADDRANYCOUNT without an IPL.

### 17.5.1 The issue

To increase the MAXSOCKETS or INADDRANYCOUNT parm in UNIX System Services (OMVS), you need to shut down OMVS and restart it. Because many tasks use UNIX services, this process is disruptive and is better done with an IPL.

### 17.5.2 How this is addressed in z/OS 1.12

Both MAXSOCKETS and INADDRANYCOUNT can be changed using the **SETOMVS** command. Update a BPXPRMxx member with the new values and issue **SETOMVS RESET=(xx)**. Update the BPXPRMxx member used at IPL to make this change permanent.

### 17.5.3 Restrictions or considerations

Care needs to be taken that other changes to OMVS are not accidentally introduced by issuing the SETOMVS command against an out-of-date member.

## 17.5.4 Rollback

This has not been rolled back to previous releases.

## 17.5.5 Further information

The **SETOMVS** command is documented in: *z/OS MVS System Commands*, SA38-0666. This includes a table of OMVS values that can be modified.

For more information, see *UNIX System Services Planning*, GA32-0884.

## 17.6 Remount with same mount mode

This section covers an enhancement that allows you to remount the system in the current mode.

### 17.6.1 The issue

There were numerous I/O errors on your file system and it has become unusable. You need to unmount the file system and remount it, but other file systems are mounted on it.

### 17.6.2 How this is addressed in z/OS 1.11

From the UNIX System Services environment, a user issues the **chmount -s pathname** command. This command remounts the file system in the current mode.

### 17.6.3 Restrictions or considerations

To issue the command, the user requires either uid(0) or READ access to the Resource Access Control Facility (RACF) resource SUPERUSER.FILESYS.MOUNT under the UNIXPRIV class.

### 17.6.4 Rollback

This command has not been rolled back to previous releases.

### 17.6.5 Further information

For more information about the command, see *z/OS UNIX System Services Command Reference*, SA22-7802.

For more information about the RACF UNIXPRIV entry, SUPERUSER.FILESYS.MOUNT, see *UNIX System Services Planning*, GA32-0884.



## Part 2

# z/OS middleware

This part of the book describes middleware components that run on z/OS and how they can contribute to your ability to plan outages. These include CICS, IMS, DB2, and WebSphere MQ.

This part includes the following chapters:

- ▶ CICS considerations
- ▶ DB2 considerations
- ▶ IMS considerations
- ▶ WebSphere MQ for z/OS





# CICS considerations

This chapter describes new facilities and techniques in recent releases of CICS that reduce or eliminate the need to perform a planned recycle of individual CICS regions. It also covers enhancements in a Parallel Sysplex Environment to maintain application availability even if individual CICS regions need to be restarted.

Although information in this chapter might be relevant to reducing unplanned loss of CICS regions or using Parallel Sysplex, it is not intended to provide detailed or comprehensive information in these areas. There are many sources for information for these areas, of which the following list is a small sample:

- ▶ *System z Parallel Sysplex Best Practices*, SG24-7817
- ▶ *System z Parallel Planning for CICS Continuous Availability*, SG24-4593
- ▶ *Parallel Sysplex Application Considerations*, SG24-6523
- ▶ *Threadsafe Considerations for CICS*, SG24-6351

This chapter includes the following sections:

- ▶ Possible reasons to schedule stopping a CICS region
- ▶ Techniques to reduce unavailability through configuration changes
- ▶ Reduction of preventive recycles of CICS
- ▶ Reducing shutdowns through runtime dependencies
- ▶ Reducing shutdowns to allow resource access by non-CICS tasks
- ▶ CICS still must be stopped, so what next?

## 18.1 Possible reasons to schedule stopping a CICS region

Although there are many reasons for stopping a CICS region, most come under one of the following categories:

- Configuration changes

Examples include changes to the job control language (JCL) for the region, changing resources defined in the CICS region, data moves, and using configuration changes made in the z/OS system. These include changes to the system clock, updated RACF key ring files, and MVS user ID changes.

- Preventive

There might be a known or perceived problem that causes the region to become unusable if the environment is not restarted at regular intervals. Examples include an application that experiences storage leaks or logs becoming full.

- Runtime dependencies

If an external resource used by a CICS region needs to be made unavailable, CICS regions might also need to be stopped. For example, if a DB2 subsystem that CICS is connected to becomes unavailable, the connected CICS regions produce errors when any CICS applications try to use DB2, so you might think that the CICS regions must be stopped if DB2 is being stopped.

- Use of resources by non-CICS tasks

The most common example of this is that files that are being accessed by CICS need to be backed up or accessed by batch jobs.

Regions might also be stopped regularly for historical reasons that are no longer applicable. Reviewing the rationale might mean that the number of planned outages can be reduced with no actual changes in the environment's configuration. Give consideration to whether a CICS region being available, but with certain facilities and resources temporarily unavailable, is sufficient as this might be much simpler to implement than trying to achieve complete availability. For example, does it matter if a printer is unavailable when the office it is in is unstaffed?

## 18.2 Techniques to reduce unavailability through configuration changes

As mentioned in 18.1, "Possible reasons to schedule stopping a CICS region" on page 184, configuration changes can include changes to the JCL of the CICS region, the configuration of the CICS region, resources defined in the CICS region, reorganization of data, and changes to the underlying z/OS environment.

### 18.2.1 Reloading programs

Many experienced CICS system programmers are already aware that it is possible to reload an updated program from the program libraries in use. However, given the abundance of features and facilities in CICS, it is not unusual for capabilities that are familiar to some to be unknown by others.

After CICS has loaded a program into memory for execution, it is not deleted from memory when execution is complete. Instead, it is kept and reused for further executions of the program.

**Remember:** There are a few specific circumstances when this does not occur. Detailing them is beyond the scope of this book.

This removes the processor cycles needed to load the program again for future executions of the program. However, it can lead to the misconception that new versions of the program in the load library are not used by a running CICS region.

Since at least CICS/ESA Version 3 Release 1 Modification 1 around 1993, CICS can be instructed to start using a new version of the program through use of the **exec cics set program** command specifying **newcopy** or **phasein** values for the option **copy**. Assuming that the command runs successfully, any future executions of the program use the updated version of the program. The key difference between the two options is that **phasein** means that any currently running instances of the program, or tasks with the program lower in their execution stack, continue with the old version and CICS deletes it from memory when no longer in use. The **newcopy** option fails if the program is in use. For more information about the **exec cics set program** command, see the *CICS Transaction Server for z/OS Version 5 Release 1 System Programming Reference*, SC34-2872.

**Note:** If the CICS program libraries have been changed, the program might be reloaded from a different data set. For more information, see 18.2.3, “Dynamic program library management” on page 186.

## 18.2.2 Partitioned Data Set Load Library versus Extended

Originally, program libraries had to be defined as a partitioned data set (PDS), which has several limitations:

- ▶ Space is not reclaimed when members are deleted, a new version of a member uses extra space within the PDS, and space that was used by previous versions are still in the PDS. Consequentially updates can potentially cause the data set to grow into more extents.
- ▶ If a PDS starts using a new extent, existing users, such as CICS, are unable to retrieve members in this extent, which causes I/O errors.
- ▶ Maximum of 16 extents.
- ▶ Space can only be reclaimed by compressing the data set, which requires exclusive access.

**Note:** A PDS can be compressed in shared mode, but carries the risk that users might not detect a member has moved within the data set.

- ▶ Directory of a PDS has a specified size at allocation. If a PDS directory becomes full, it must be reallocated to increase the directory size.

First introduced in 1989, partitioned data set extended (PDSE) relieves many of these limitations:

- ▶ Space is reclaimed automatically, reducing the potential to grow into more extents.
- ▶ CICS does not require a recycle to retrieve members from a new extent.
- ▶ Maximum of 123 extents.
- ▶ Directory expands automatically as needed.

For more information about PDSEs, see *Partitioned Data Set Extended Usage Guide*, SG24-6106, available at:

<http://www.redbooks.ibm.com/abstracts/sg246106.html?Open>

### 18.2.3 Dynamic program library management

Before CICS Transaction Server Version 3 Release 2, whenever CICS loaded an application program, whether in response to an explicit load instruction or an implicit load such as a CICS transaction starting to run the program, the CICS Loader Domain searched through the list of data sets specified in the DFHRPL concatenation in the region's JCL and loaded the first instance of the program.

The LIBRARY resource introduced in CICS Transaction Server Version 3 Release 2 added the capability to define up to 16 data sets as a group that can be added or removed from the region to be searched before or after the DFHRPL concatenation. If a LIBRARY installed in a region has a status of DISABLED, it is no longer allocated to the region, enabling any activities that require exclusive access to the load libraries to be performed. When ENABLED, the library data sets are dynamically allocated by using a data definition name (DD Name) matching the name of the LIBRARY.

Typical examples of the usage of CICS LIBRARY resources include adding emergency fixes to a program, enabling it to be identifiable as such and grouping the load libraries for a particular application, possibly using different LIBRARY names to identify different versions. For more information about dynamic program library management, see *CICS Transaction Server for z/OS Version 5 Release 1 Application Programming Guide*, SC34-2844.

### 18.2.4 Dynamic allocation of files and Extrapartition Transient Data Queues

Earlier versions of CICS required that any data sets that CICS was required to access as VSAM files or Extrapartition Transient Data Queues (sometimes called "Extra TDQs") had to be defined directly in the JCL to be usable by CICS. Although this is still a valid method, the data sets remain allocated to CICS even if they are closed, preventing other non-CICS based activities against the data set. The degree of the restriction depends on factors such as the disposition specified in the JCL, and the level of access required to perform the non-CICS activity.

Current versions of CICS support specifying the data set name on the CICS resource definition. This causes CICS to dynamically allocate the file when opened by CICS, and to deallocate the file when closed. For VSAM files, the DD Name used in the allocation is the same as the CICS File name. For Extra TDQs, the DD Name used is specified in the CICS TDQueue resource definition. Additionally, the use of dynamic allocation for files and Extra TDQs allows you to change the data set in use dynamically by closing and disabling the file or queue. You can then install a new or modified definition of the same name. For VSAM files, this can also be achieved more simply by using the **exec cics set file dsname()** command.

If using dynamic allocation for files or TDQs, the following points might be of use:

- ▶ When installing a dynamic allocation resource, if the DD Name is already in use in the JCL or an already in-use dynamic resource, the entry in the new definition will be overridden to the existing one.
- ▶ CICS does not permit the specification of member names in a dynamically allocated TDQ definition.

- ▶ Extra TDQs specified in JCL used as input can use a concatenation of data sets. But dynamically allocated ones can have only a single data set specified.
- ▶ CICS will not allow a TDQ whose name starts with the character C to be discarded, but CICS files and TDQs that are closed and disabled can be replaced by installing the new definition without discarding, provided that the type of TDQ is the same as that already present.

For more information about defining dynamic FILEs and TDQs, see *CICS Transaction Server for z/OS Version 5 Release 1 Resource Definition Guide*, SC34-2848

For more information about the **exec cics set file** and **exec cics set tdqueue** commands, see *CICS Transaction Server for z/OS Version 5 Release 1 System Programming Reference*, SC34-2872.

## 18.2.5 Changes to CICS System Initialization Table values.

The CICS system initialization table (SIT) specifies most of the parameters that pertain to the CICS region. It controls whether certain facilities are available in the region, how many user tasks the region can run, how much storage is available, and so on. These parameters can be supplied to CICS in several ways:

- ▶ As a loaded table, defined using the DFHSIT assembly language macro and assembled into an APF-authorized library in the CICS STEPLIB DD Name concatenation.
- ▶ By using the PARM option on the EXEC statement in the CICS JCL.
- ▶ As values read from SYSIN DD Name concatenation of the CICS JCL.
- ▶ As operator-supplied values from the console.

**Note:** Certain SIT parameters, primarily those related to security settings, cannot be specified through the console.

Not all SIT values can be changed dynamically, meaning a restart of CICS is required to change those values. Careful consideration should be given to initial values.

### Information about CICS retention of SIT values over restarts

When performing a COLD or INITIAL start, CICS reloads all SIT values from the sources in the following order:

1. Assembled SIT table
2. PARM statement in the CICS JCL
3. SYSIN DD Name (only if **SYSIN** is coded in the PARM statement).
4. Operator supplied input (only if **CONSOLE** or **CN** is coded in PARM or SYSIN).

#### **Steps on a COLD or INITIAL start**

The first parameters to be searched for are **SYSIN** and **CONSOLE** to determine where any SIT overrides are to be retrieved from, followed by **SIT=xx** with CICS taking the last value supplied to determine the assembled DFHSITxx to load.

Having loaded the SIT, any supplied overrides from PARM, SYSIN, and CONSOLE are used to change the loaded values with last specified value used should multiple occurrences be found.

The SIT values in use are stored in the CICS Global Catalog Data Set (GCD) when resolved, dynamically changed, and during CICS shutdown for reuse during a Warm or Emergency restart.

### Steps on a Warm or Emergency start

The Warm or Emergency start sequence is similar to a COLD or INITIAL start. The exception is that unless override **NEWSIT=YES** has been specified or the value for **SIT=xx** has been changed, CICS retrieves the SIT values stored in the CICS GCD by the previous instance and *applies any values specified as overrides*. This behavior can cause confusion. If you do not know that a SIT value has been specified as an override, CICS appears to be resetting some SIT values on a restart and restoring others in an unpredictable way.

**Note:** Parameters that are related to CICS tracing are not stored in the CICS GCD and are always reestablished from the SIT and SIT overrides on a restart with any dynamic changes discarded.

### Dynamically changing SIT values

While a CICS region is running, some values specified by a SIT parameter might require changing for various reasons. Although, as already mentioned, many cannot be changed dynamically, many that are most likely to require dynamic change can be manipulated using SPI commands, some of which are listed in Table 18-1.

Table 18-1 Examples of SIT Parameters that can be updated dynamically

SIT Parameter	Brief Description	CICS SPI to manipulate	Notes
AUXTR	Turn auxiliary trace on and off	<b>SET TRACEDEST</b> or transaction <b>CETR</b>	
AUXTRSW	Control auxiliary trace file switching		
AKPFREQ	Number of writes to CICS system log before taking a keypoint	<b>SET SYSTEM</b>	
DEBUGTOOL	Use Debugging Tools		
[E]DSALIM	Control maximum amount of storage used by CICS in 24-bit and 32-bit storage		
DUMP	Whether CICS will take System memory dumps		
DUMPDS	Transaction memory dump Data set to use	<b>SET DUMPDS</b>	
DUMPSW	Whether to switch data sets when Transaction memory dump Data Set is full		
INTTR[xx]	Level of Standard tracing for CICS Components	<b>SET TRACEFLAG</b> or transaction <b>CETR</b>	

SIT Parameter	Brief Description	CICS SPI to manipulate	Notes
MAXSOCKETS	Maximum of TCP/IP Sockets to be used by the CICS region	<b>SET TCPIP</b>	
MXT	Maximum number of tasks in the CICS System	<b>SET SYSTEM</b>	
SPCTR[xx]	Level of Special tracing for CICS Components	<b>SET TRACETYPE</b> or transaction <b>CETR</b>	
SYSTR	Switch CICS System tracing on or off	<b>SET TRACEFLAG</b> or transaction <b>CETR</b>	
TRTABSZ	Size of internal trace table held in memory by CICS	<b>SET TRACEDEST</b> or transaction <b>CETR</b>	
TSMAINLIMIT	Amount of storage above 2 GB used by main Temporary Storage	<b>SET TEMPSTORAGE</b>	
USERTR	Switch CICS User tracing on or off	<b>SET TRACEFLAG</b> or transaction <b>CETR</b>	

## 18.2.6 Reload an updated RACF Certificate key ring

Before CICS Transaction Server Version 5 Release 1, CICS loaded the key ring specified by the SIT parameter `key ring` at initialization. Any changes to the certificates connected to the key ring do not take effect in an individual CICS region until the region was restarted. This potentially significant delay in using key ring updates in applications or CICS configurations has been removed with the **exec cics perform sslrebuild** command. For more information, see *CICS Transaction Server for z/OS Version 5 Release 1 System Programming Reference*, SC34-2872.

**Restriction:** If the specified key ring does not exist in the RACF database when CICS is started, CICS reports the value as invalid. Depending on the setting of SIT parameter `PARMERR`, this can either halt CICS start or cause CICS to not load any key ring.

**Note:** If a CICS region is started specifying a key ring that is empty, errors will be encountered during CICS initialization.

## 18.2.7 Change of Local Time on z/OS system

To match the ability of z/OS to dynamically change the time (normally used for Daylight Saving Time), CICS can be instructed to resynchronize with the Multiple Virtual Storage (MVS) time-of-day (TOD) clock using **exec cics perform resettime**.

For releases before CICS Transaction Server Version 5 Release 1, the default value of SIT parameter `AUTORESTTIME` is `NO`. This causes CICS to compare its internal TOD clock with the

MVS one at midnight local time and issue message **DFHAP1500** to indicate that the CICS and MVS clocks differ by 30 minutes or more.

CICS Transaction Server Version 5 Release 1 changes the default value of **AUTORESETTIME** to **IMMEDIATE**, which causes CICS to compare the clocks whenever a transaction is started and resynchronize if they differ. This check will be triggered by a terminal connecting to CICS as this runs CICS internal transactions and the “Good Morning” transaction. Unlike **AUTORESETTIME=YES**, resynchronization will occur for any difference not just thirty minutes or more.

The immediate option was also added to existing releases of CICS by the APARs released in 2012 as shown in Table 18-2. Although not the default value for Versions 3 and 4, select the value of **IMMEDIATE**.

Table 18-2 APARs extending **AUTORESETTIME** to include **IMMEDIATE**

CICS Transaction Server Versions	APAR	PTF
3.1	PM73143	UK83664
3.2	PM52172	UK78322
4.1	PM52109	UK77263
4.2	PM61466	UK78430

### Items to consider when dynamically resynchronizing the CICS clock

As already mentioned, dynamically changing the CICS clock avoids the need to recycle a CICS region when the local time changes. However, depending on what use is made of the local time within the region and its applications, changing the CICS time dynamically can result in unintended effects. The following sections detail some of the more common issues that can occur. More information about the commands can be found in *CICS Transaction Server for z/OS Version 5 Release 1 Application Programming Reference*, SC34-2845.

#### Application generated time stamps

If an application generates time stamps, for instance by using CICS commands **asktime** and **formattime**, a change in the local time for Daylight Saving Time (DST) *forwards* causes the time stamps generated to advance an hour. The implications of this depend on how this information is used. For example, tasks might appear to take an hour longer than they actually took, or there appeared to be no activity at all on the system for an hour. The implications when DST causes the clock to *go back an hour* can potentially be more serious. For example, transactions can appear to have finished before starting, giving negative execution times, time stamps in sequential logs appear to be out of sequence, or the number of tasks that are run appears to be roughly double the normal level for an hour.

Whether these considerations are an issue is an individual choice, the answer to which varies from environment to environment, and possibly even between different applications. There are multiple techniques to deal with any issues, the suitability of which, as with any application change, is a balance between the significance of the issue against the cost of application changes. The following are potential methods to handle changing times:

- Write log records to an MVS System Logger logstream by using the **exec cics write journalname** command. These records are timestamped in Greenwich mean time (GMT) by System Logger, so they might not require a time stamp to be generated by the application.
- Generate time stamps in a universal time zone such as GMT or Coordinated Universal Time (UTC) using the **datestring** and **stringformat** options of **exec cics formattime**.



These newer options, introduced in CICS Transaction Server Version 3 Release 1, can also be converted to the current local time using `exec cics converttime` or custom application code.

**Note:** `exec cics converttime` gives an `abstime` value for subsequent use by `exec cics formattime`.

- Depending on individual application needs, it might be acceptable to dynamically advance the time for DST, but stop the environment for an hour when the DST causes the clocks to regress an hour.

### ***Interval Control Commands***

CICS has several interval control commands that enable tasks to delay execution or schedule some activity in the future. When using these commands, CICS provides two alternatives to specify when the delay is to end or activity is to occur: As an interval (how long after the command is issued) or as a specific time.

Intuitively, these alternatives are the same. A command that is issued at 1:05 am to start a task in an hour seems the same as a command issued at the same time to start a task at 2:05 am. However, if the clock is set back an hour, the first will still wait an hour ending at 1:05 am (new local time) while the second waits until 2:05 am (new local time), an elapsed time of two hours. Conversely, had the clock been set forward an hour, the first still waits an hour until 3:05 am while the second stops waiting as soon as the CICS time changes. Depending on why the application is being started in the future, either behavior can be the wanted approach.

**Note:** After and At in Interval Control commands behave differently on clock changes.

A potential unexpected consequence of specifying a specific time can be that a clock change forward might cause the region to become flooded with tasks as multiple starts are triggered immediately rather than spread out over time.

### ***SMF records***

The System Management Facilities (SMF) header records contain the local time the record was written with a field in the CICS monitoring data called `SMFMNDTO` holding the local offset. However, the header is generated by SMF and so contains the actual local time of the system while the local offset is written by CICS. This means if the region has not resynchronized clocks after a clock change (see 18.2.7, “Change of Local Time on z/OS system” on page 189) the two time-related fields in the record can be inconsistent.

CICS statistics records are written out when the local time of the region either reaches the value specified in the `STATEOD` SIT parameter or the time since the last reset reached the `STATINT` SIT value. However, if the clock advances an hour, the statistics records will be for an hour less than specified by `STATINT`, but are still reported by utilities such as `DFHSTUP` as the correct interval.

## **18.2.8 RACF changes**

After CICS has retrieved the information for a user ID from Resource Access Control Facility (RACF), the details are held in a CICS internal cache unless no tasks are run for this user for a period of time equal to the SIT parameter `USRDELAY`. Before CICS Transaction Server Version 4 Release 1, CICS would be unaware of RACF changes to the user ID unless a signon was attempted for the user ID that resulted in RACF reporting the ID was revoked.

This lack of awareness meant that CICS might indefinitely continue running tasks for a revoked user or allow access to resources that are no longer permitted.

Although this can be mitigated by configuring a region with **USRDELAY** set to zero or a low value, this causes increased processor burden with CICS requesting information from RACF more frequently. Unless set to zero, it might still allow use of revoked user IDs if used frequently enough so the only two options to ensure that CICS reflects all RACF changes were to endure the burden of **USRDELAY=0** or periodically recycle CICS.

CICS Transaction Server Version 4 Release 1 added the capability of CICS to listen for Event Notification Facility (ENF) events type 71 issued by RACF. On receipt of an ENF 71, CICS marks the cache entry of the affected users as invalid so any terminal or remote task that starts or issues a sync point request under a such a user uses refreshed RACF information.

CICS receives a notification for the following RACF commands:

- ▶ z/OS 1.11 or later: ALTUSER with the REVOKE option, CONNECT, and REMOVE.
- ▶ z/OS 1.13 with the PTF for APAR OA39486 applied, or later: ALTUSER with the REVOKE option, CONNECT, REMOVE, DELGROUP, and DELUSER.

For more information, see *CICS Transaction Server for z/OS Version 5 Release 1 RACF Security Guide*, SC34-2866

## 18.2.9 Tracking dynamic changes to a CICS region

Before CICS Transaction Server Version 5 Release 1, tracking dynamic changes to a CICS region was difficult without rigorously followed procedures to track changes. In CICS Transaction Server Version 5 Release 1, this was made potentially easier by the addition of audit messages of SPI commands potentially causing changes to the system. Example 18-1 shows examples of these new messages, which contain the local data and time, CICS VTAM ID, the terminal identifier (if any), user ID, transaction, command issued, and outcome.

*Example 18-1 Sample DFHAP1900 audit messages*

---

```
DFHAP1900 07/25/2013 15:18:39 CIQ3GT TCPSA2 CLIFTON CETR SET TRACEDEST
TABLESIZE(30000) RESP(NORMAL) RESP2(0).
DFHAP1900 07/25/2013 15:27:26 CIQ3GT NONE CLIFT02 CONL SET PROGRAM(CEECBLDY)
COPY(PHASEIN) RESP(NORMAL) RESP2(0).
```

---

## 18.2.10 Applying Service

To use a new service or upgrade the level of CICS being run, a region must be restarted. Although updated CICS modules can be copied into the CICS product libraries without restarting, this carries a high risk of regions becoming unstable as changes can affect multiple modules or change entry points within modules.

For example, an update might change an internal interface between two CICS modules so a running region that already has both loaded will not be using the new service. However, should it only have one module loaded and load the second later through normal CICS program management activity, the two loaded modules will have different information about how they communicate.

However, although a CICS region must be restarted to apply service, the time it is unavailable can be minimized by using an environment allowing as much as possible to be performed before any CICS region must be stopped.

## Separate SMPE and execution libraries

Using SMPE to apply service to a different set of libraries than those in use by the running CICS regions enables service to be applied to the environment without initially impacting running systems. After any required preparation to adopt the new service is complete, regions can then be shut down, the updated service copied into the libraries used by the running CICS regions. These regions which can then be restarted. For more information about how this approach can be expanded further for increased flexibility, see “Using symbols in JCL and data set aliases” on page 193 and “Rolling updates” on page 195.

**Note:** When using this approach, attention needs to be paid to any installation processes that further configure product libraries and their contents to match the specific environment. For example, if modules are bound to a particular DB2 database or CICS regions, recopying the SMPE data sets might require them to be bound again.

## Offline updates

Occasionally, the service being applied requires other updates to the environment to use the changes supplied, such as upgrading the CICS system definition (CSD) data set to add new or changed definitions or recompile programs to reflect copybook changes. These changes can normally be undertaken in advance of shutting down regions, and program recompiles might not be required depending what has changed. For example, if a copybook is updated to include a new field, it is unlikely that an existing application program will be using the new field. Therefore, that application will not need to be recompiled if the layout of fields that are being used is unchanged.

## Using symbols in JCL and data set aliases

Using multiple product library data sets has already been mentioned in “Separate SMPE and execution libraries” on page 193, although there must be some method to determine which set of libraries to use.

Specifying the exact set of libraries to use in the JCL is possible. However, this process represents a large amount of processor burden every time maintenance is applied, and has a significant potential for error with multiple changes required in each job. This becomes more complicated if different systems are using different product libraries, perhaps because updates are being applied on an image by image basis.

Using symbols enables the JCL to remain unchanged but still use the appropriate libraries. Example 18-2 shows symbols being used to determine both product high-level qualifiers and the service level in use. The double period is because the first indicates the end of the variable name and the second period is required in the final data set.

**Note:** Although not shown in the examples here, symbols can be used anywhere in JCL to specify required values such as the volume holding the required data set.

### *Example 18-2 Use of symbols in JCL*

---

```
//STEPLIB DD DSN=&CICSHLVL..&SRVLVL..CICS.SDFHAUTH,DISP=SHR
//          DD DSN=&CICSHLVL..&SRVLVL..CICS.SDFJAUTH,DISP=SHR
//          DD DSN=&CPSMHLVL..&SRVLVL..CPSM.SEYUAUTH,DISP=SHR
//          DD DSN=&MQSYS1..&SRVLVL..SCSQANLE,DISP=SHR
//          DD DSN=&MQSYS1..&SRVLVL..SCSQAUTH,DISP=SHR
```

---

When JCL containing symbols is run, the output shows the substitution that has taken place to indicate the actual data set used as shown in Example 18-3.

*Example 18-3 Resolution of variables in job output*

---

```
XXSTEPLIB DD DSN=&CICSHLVL..&SRVLVL..CICS.SDFHAUTH,DISP=SHR
IEFC653I SUBSTITUTION JCL - DSN=CICSTS41.RSU1303.CICS.SDFHAUTH,DISP=SHR
XX DD DSN=&CICSHLVL..&SRVLVL..CICS.SDFJAUTH,DISP=SHR
IEFC653I SUBSTITUTION JCL - DSN=CICSTS41.RSU1303.CICS.SDFJAUTH,DISP=SHR
XX DD DSN=&CPSMHLVL..&SRVLVL..CPSM.SEYUAUTH,DISP=SHR
IEFC653I SUBSTITUTION JCL - DSN=CICSTS42.RSU1303.CPSM.SEYUAUTH,DISP=SHR
XX DD DSN=&MQSYS1..&SRVLVL..SCSQANLE,DISP=SHR
IEFC653I SUBSTITUTION JCL - DSN=SYS1.MQ710.RSU1303.SCSQANLE,DISP=SHR
XX DD DSN=&MQSYS1..&SRVLVL..SCSQAUTH,DISP=SHR
IEFC653I SUBSTITUTION JCL - DSN=SYS1.MQ710.RSU1303.SCSQAUTH,DISP=SHR
```

---

Symbols can either be resolved using System Symbols or by setting them explicitly within the batch job (or more likely an included JCL member). Given that symbolic variables are only resolved for started tasks and not for batch jobs before z/OS 2.1, many sites use standard JCL include files held in image-specific procedure libraries to set the required symbols on that image. For more information about using System and JCL Symbols, see the *z/OS MVS JCL Reference*, SA23-1385.

Although symbols can be used to determine the correct data set to use for jobs, it does not help when using facilities, such as ISPF, to examine copybooks. Using data set aliases might be a more practical option.

A data set alias is an MVS catalog entry that provides a redirection to the real data set name rather than indicating the volume that a data set is physically on. Two forms of alias are available: A direct relationship between an alias and its target, and a symbolic relationship that enables system symbols to be used in the specification of the target. Example 18-4 shows commands defining an alias using a direct relation and a symbolic relation.

*Example 18-4 Examples of commands to define an alias data set*

---

```
DEFINE ALIAS ( NAME(REGTEST.UTIL2.LOCAL.CLIST) -
               RELATE(REGTEST.UTIL2.HURPLEX2.CLIST))

DEFINE ALIAS ( NAME(REGTEST.UTIL.LOCAL.CLIST) -
               SYMBOLICRELATE(REGTEST.UTIL.&SYSPLEX..CLIST))
```

---

Because an alias is a catalog redirection, it can be modified without effecting any running tasks that have related data sets open. Any change to the alias either explicitly or through dynamic change of a system symbol will take effect the next time a task allocates and opens the data set using the alias.

**Note:** An alias using a symbolic relationship can have different target data sets on different systems within a sysplex because system symbols are unique to the individual system. The same JCL can be submitted to different systems and transparently use different data sets on each. However, this configuration can cause confusion if you are looking at data sets on one system being accessed by a job on a different image.

**Attention:** Remember that a data set specified in JCL is allocated at the time the job or job step starts running, so any change in alias definition has no effect on the running job. However, if the file is dynamically allocated through 18.2.3, “Dynamic program library management” on page 186 or 18.2.4, “Dynamic allocation of files and Extrapartition Transient Data Queues” on page 186, alias changes take effect as soon as the resource is deallocated and reallocated.

An alias must be in the same catalog as the data sets it redirects to.

### **Allocate a replacement on a different volume**

An alternative technique is to define updated versions of data sets using the same name as its replacement but on a different volume, and then change the catalog entry to point to it. You can also specify the volume in the JCL, preferably using a symbol. The effect is the same as changing an alias: New allocate requests use the new data set whereas existing users continue with the old version.

This approach can only be used for data sets that are not managed by SMS. Care must be taken to track old versions of data sets to prevent DASD space being lost to unidentified orphan data sets.

### **Rolling updates**

Rolling updates is a technique that is often used in large environments to minimize the time that users perceive that a process takes by applying service to smaller parts of the system at a time. The overall time that it takes to apply service to the entire environment might be several days or longer depending on the size of the environment and processes being followed. However, each individual region is unavailable for a short period. For more information about increasing the users’ perceived availability, see 18.6, “CICS still must be stopped, so what next?” on page 214.

Although it is rare for CICS Service to require an MVS image to be go through IPL, other services including hardware maintenance might do so. In these circumstances, these techniques to reduce unavailability might still help prepare for the image being restarted to reduce the time that is taken to restart the regions afterward.

## **18.2.11 Upgrading CICS**

When upgrading the level of CICS, a region must be shut down. This section provides techniques to reduce the time that a region is unavailable.

Upgrading the version of CICS is a significant undertaking that requires careful review of the CICS documentation detailing what has been changed. This process helps to determine the steps that are required to prepare the environment to run the new level. In general terms, the upgrade process can be seen as two distinct steps:

1. Install the new level of code in the system.
2. Upgrade the CICS regions to the new level.

Before CICS Transaction Server Version 5 Release 1, an image always required an IPL to install the new level of the CICS SVC type 3. However, this need has been removed with the provision of a CICS utility DFHCSVCU, which dynamically updates or adds it. However, the CICS upgrading documentation also contains the following warnings:

- ▶ For an existing SVC, before the SVC is updated all regions using the number must be shut down. Otherwise, the results can be unpredictable.
- ▶ Changes made by DFHCSVCU are temporary, so the system configuration must also be updated. Otherwise changes will be lost when system goes through IPL.
- ▶ If using multiregion operation (MRO) to communicate between regions, all regions must use the highest level type 3 SVC.
- ▶ When upgrading DFHIRP (used by MRO), all users must stop using it. Either close CICS interregion communication (IRC) or shut down the region.

After it is installed in an image, the upgraded CICS link list and link pack area (LPA) modules are written to be usable by any earlier version of CICS without change. However, perform testing appropriate to the local procedures.

Before upgrading individual CICS regions to the newly installed version, further checking for changes in documentation can save significant amounts of time diagnosing problems when trying to start the regions. Here are a few questions for your checklist:

- ▶ Have any facilities been removed?  
A removed CICS facility might mean that some definitions and groups supplied by CICS have also been removed. If sharing the CSD between CICS releases, the old versions that might need these definitions will have been moved to a **DFHCOMPATx** group. If you are using a private group list, that list might contain deleted groups.
- ▶ Have any SIT definitions been changed?  
Some changes might have further consequences. For example, **TSMINLIMIT** cannot be more than a quarter of the value of **MEMLIMIT**, so an increase in one might require an increase in another. This example is not meant to imply that **TSMINLIMIT** is known to have changed.
- ▶ Have any minimum requirements changed for CICS itself?  
CICS Transaction Server Version 5 Release 1 increased the minimum value for **MEMLIMIT** to 6 GB from 4 GB in CICS Transaction Server Version 4 Release 2, or 2 GB in CICS Transaction Server Version 4 Release 1. Too low a value causes CICS to issue an error message during initialization, take a system memory dump, and terminate.
- ▶ Any internal changes in CICS that impact requirements to run the region or system?  
CICS Transaction Server Version 4 Release 2 moved the internal trace table and main temporary storage in 64-bit storage (above the bar) which was followed by CICS Transaction Server Version 5 Release 1 moving stored containers to join them. While making more 31-bit (above the line) storage available in a running region, this can increase the amount of storage required above the bar. This in turn means that any system memory dumps will be larger, requiring more space to capture and store memory dumps. This affects both the memory used by **DUMPSRV** to capture and disk space to store.
- ▶ Do any applications or tools need changing?  
Although it is unusual for application programs to require changes to run on an upgraded level of CICS, it should never be assumed. In particular, tools and third-party applications might check that the level of CICS is one they support, or might be use CICS exit points and user replaceable modules that have changed interfaces.

As mentioned in “Rolling updates” on page 195, the checks mentioned here help reduce the time that a region is unavailable to be upgraded. The items that are described in 18.6, “CICS still must be stopped, so what next?” on page 214 might further reduce the user’s perception of how long the application is unavailable.

## 18.3 Reduction of preventive recycles of CICS

In some circumstances, a CICS region might be recycled regularly to prevent some cumulative problem becoming so severe that the region becomes unstable or unresponsive. Obviously there are many reasons that might be specific to an individual environment. In this section, a few examples of problems and potential root causes are described to illustrate some tools and techniques to identify the cause or detect the problem developing early enough to enable less disruptive preventive action to be taken. Depending on the nature of the problem, some actions can be automated.

As mentioned in 18.1, “Possible reasons to schedule stopping a CICS region” on page 184, regular review of preventive recycles and their underlying reasons might reveal the issues have been resolved. Even if the recycles are still required, it might be that their frequency can be reduced.

### 18.3.1 Storage shortage

The most likely reason a region might be regularly recycled is to prevent storage shortages from occurring. These are usually caused by some form of storage leak rather than a peak in storage requirements, but other causes of storage shortage are also mentioned here to illustrate tools or facilities.

Storage is allocated by CICS from one of several dynamic storage areas (DSAs) whose names indicate the location of the storage and the type of storage allocated as indicated below. Not all combinations exist as valid dynamic storage areas.

- ▶ Character 1: Location of storage
  - Blank: 24 bit addressed storage below the 16 MB line
  - E: Extended storage, 31-bit addressed storage above the 16 MB line but below the 2 GB line.
  - G: “Grande” storage, 64-bit addressed storage above 2 GB
- ▶ Character 2: Type of storage
  - C: CICS storage that is protected from update by tasks not running in CICS-key
  - R: Read-only storage that is protected from update by tasks other than specific internal CICS tasks. This type is normally used to store re-entrant programs.
  - S: Shared storage that is not seen by CICS as being associated with a particular task.
  - T: Trusted storage that holds sensitive information not accessible by CICS system or user tasks except through explicit internal CICS calls.
  - U: User storage that can generally be updated by any CICS task.
- ▶ Characters 3, 4, and 5: Fixed to DSA to indicate that the area is a dynamic storage area.

Within a CICS region, the size of each individual DSA is dynamically controlled depending on the current allocated sizes, although the maximum total storage allocated in each range is controlled as follows. Some can be set to a fixed size though SIT parameters, but this requires careful consideration.

- ▶ Below 16 MB line: SIT parameter DSALIM
- ▶ Below 2 GB but above the 16 MB line: SIT parameter EDSALIM
- ▶ Above 2 GB: JCL parameter MEMLIMIT.

## Identifying storage leaks

Storage leaks can occur in a region for various reasons, including use of shared storage, programs defined as reload(yes), long running tasks not managing storage efficiently, growing numbers of tasks, or errors in the CICS code itself. As CICS itself does not report growing storage use, detecting ongoing storage issues requires proactive monitoring of the region.

The current size of each DSA can be determined through use of the **exec cics inquire system** command, which reports the size regardless whether it is allocated or available to be allocated. You can also use the **exec cics collect statistics** command, which enables more information about each DSA to be determined. This information includes how much space is free, the largest contiguous free space, and the numbers of getmains and freemains.

Although it is normal for the sizes of the various DSAs to fluctuate, the number of getmains growing faster than freemains within a DSA is a good indication of a storage leak. Which DSA is experiencing the leak can give clues to the root cause.

Environments using tools such as IBM CICSplex® System Manager (CPSM) or IBM Tivoli OMEGAMON® XE for Messaging are able to examine this information, and provide more details to help with diagnosing. CPSM can be used to monitor the amount of space available in the various DSAs, and issue warnings or take action if it is determined to be too low. The amount of storage that is used by individual tasks can also be monitored to detect heavy users of storage. This early detection can enable earlier diagnosis so you can take preventive measures taken before the region is adversely effected.

## Shared storage leaks

Shared storage is allocated by using the **shared** option on an **exec cics getmain** command. Although this has advantages, such as to enable different CICS tasks to share data directly in memory either as concurrent tasks or one task generating data for use by a subsequent task, it will remain allocated until explicitly freed by using an **exec cics freemain** command or the region is terminated. This means that it has the obvious requirement that the last task to use the storage must release it. If for any reason this does not occur (task does not run at, fails to retrieve the required addresses, or oversight means one area is not freed) a storage leak occurs. The advent of Parallel Sysplex added another risk of the affinity between the transactions being overlooked and the transaction not running on the same CICS regions.

Shared storage is often used for tasks passing complicated data structures with variable length fields. Before the introduction of channels and containers in CICS Transaction Server Version 3 Release 1, only 32K of contiguous data can be passed directly between the tasks using CICS facilities.

**Note:** Use of facilities such as Temporary Storage enable multiple pieces of data to be passed, but each individual piece of data is still restricted to 32K.

An individual CICS task can use multiple containers to hold different parts of a complicated structure, and, by associating them with the same channel, pass them to other tasks and programs even on other systems. Applications store and retrieve data from containers using



**exec cics put container** and **exec cics get container**. Only the container data currently required needs to be held in the applications own memory.

Containers can be changed to be stored in 64-bit storage in CICS Transaction Server Version 3 Release 2 to enable more data to be stored without impacting 31-bit storage. Containers are private to an individual task and data to a new task is a copy of the data and not in the same container, so they cannot be used to share data between concurrent tasks. When a task terminates, its containers are deleted but can be passed to later CICS task by using the **exec cics start** command. If data needs to be passed from one task to another that will run at some unknown time, Business Transaction Services Container are a better option. For more information about channels and containers, see the *CICS Transaction Server for z/OS Version 5 Release 1 Application Programming Guide*, SC34-2844.

A second common use of shared storage is when enabling exits in CICS because **exec cics enable program** allows only up to 64KB of storage to be made available to the exit as global storage. It is not unusual for the enabling program to allocate a larger shared area of storage for use by the exit and store the address in the global areas directly allocated by CICS on the **exec cics enable program**. If the exit is one that is enabled and disabled regularly in a running CICS region for some reason, the task disabling the exit must either free the shared storage or have some mechanism to store the address for reuse when the exit is next enabled. An easy oversight here is updating the exit and enabling the program to enable use of a larger shared area, but not updating the disabling program to free it again.

As can be imagined, the golden rule when dealing with shared storage is to ensure that there is a mechanism to release the storage again when it is no longer required that has robust error handling. The challenge is often how to identify what programs are allocating the storage. After the program is identified, it can then be determined if it needs to be shared and how it should be released. For information about how to determine which programs require further investigation, see “DB2 Thread Use” on page 204.

### ***Programs defined RELOAD(YES)***

Although program usage itself does not intuitively seem related to storage leaks, programs that are defined as **reload(yes)** can cause storage leaks. They are intended for use with non-reentrant programs and cause CICS to load a new copy of the program each a request is made for it. However, *CICS does not release the program unless explicitly requested* by an **exec cics freemain** command by the task causing the load.

Which DSA is affected by this repeated loading of programs depends on a number of factors. When CICS loads a program from execution, where it gets loaded is governed by a number of factors. If it is linkedited with Run Mode (rmode) 24, it is loaded into storage below the 16 MB line. If marked reentrant and CICS has SIT parameter **RENTPGM=PROTECT**, it is loaded into a read-only DSA. Otherwise, programs defined **excekey(cics)** are placed in a CICS DSA and those defined **excekey(user)** are destined for a User DSA.

### ***High space usage of Main Temporary Storage Queues***

Before CICS Transaction Server Version 4 Release 2, Main Temporary Storage (Main TS) was held in the extended CICS dynamic storage area (ECDSA) with no restriction on how much storage can be used. This was then moved to 64-bit storage above 2 GB in the GCDSA and restricted in size by the SIT parameter **TSMAINLIMIT**. The current limit and storage in use can be checked by using the **tmainlimit** and **tmaininuse** options of **exec cics inquire tempstorage**.

In both instances, large amounts of data that are stored in Main TS them can cause problems with applications failing to store information in queues or the CICS region itself experiencing storage issues. Because the techniques to detect and avoid TS problems are similar

regardless of where the queue is located, see “Temporary Storage becoming full” on page 203.

### ***Deep program stacks***

Each time a program starts another, the private working storage of the current program is held to be used when control returns and extra storage is allocated for the private working storage of the newly started program. Normally, this stack of calls will not be very deep, but can be for example if an application is updated by incremental changes over time:

- ▶ A new program called PROG1 is written to attempt to run as transaction ANDY, read a record from Transient Data Queue, and process the data if any were read. When first written, the queue only ever holds a single record or is empty, so the program runs these tasks:
  - Reads the queue.
  - If a record is retrieved, process the data.
  - Terminate.
- ▶ It is now determined that additional information is required to be generated when processing the record so the code is changed to:
  - Read the queue.
  - If a record is retrieved
    - Determine how much storage required for the extra information
    - Issue **exec cics getmain**.
    - Process the data
  - Terminate.

No exec cics freemain is issued as “CICS deals with that when the task ends”.
- ▶ The program is now required to check for records every hour, and so is modified to wait for an hour and then link to itself to check again.
  - Read the queue.
  - If a record is retrieved
    - Determine how much storage required for the extra information
    - Issue **exec cics getmain**.
    - Process the data
  - **exec cics delay for hours(1)**
  - Issue **exec cics link program(PROG1)**
  - Terminate.

This last change causes two storage leaks. First, the program is now recursively calling itself, which results in an increasing stack of programs. The **rescount** option of an **exec cics inquire program(PROG1)** indicates how many times the program is being used and will be increasing by one every hour. Each entry on the recursive stack counts as another instance of being in use. Replacing the **exec cics link** with an **exec cics xctl** indicates that there is no requirement to return to the calling program and prevent the original unrequired instance’s environment from being kept.

The second leak is the storage allocated when processing a record. Because the task never ends, CICS does not free the storage requested. Even using **exec cics xctl** does not help because it is impossible to determine whether the allocated storage is still required. Changing the program from delaying an hour and then issuing the link or **xctl** to issuing **exec cics start transid(ANDY) after hours(1)** removes this task from the CICS region until next required. This process causes the storage to be released by the task termination processing. However, had the program been originally written to follow best practices and issue **exec cics freemain** for the storage when no longer required, this storage leak would not have occurred.

### ***Tasks using large amounts of storage: Policy Support***

CICS Transaction Server Version 5 Release 1 introduced *policy support* as part of the platform management enhancements that enabled CICS to monitor tasks for various activities such as the number of `exec cics getmain` requests, amount of storage used, or amount of CPU time taken. If specified thresholds are exceeded, configured actions can be performed including issue a message, abend the task, or emit a CICS event to perform tailored processing. For more information about policies, see “Cloud Enablement with Platforms” in the CICS Information Center.

### **Increasing numbers of tasks clogging the system**

Consider the scenario of CICS controlling an elevator, each time someone presses the call button or selects a floor from inside, a new task is started to send the elevator to the requested floor. During times of low usage, this might be seen as an ideal transactional task, which is a short-lived task runs to move the elevator to the required floor. Now apply this model to the start of the working day: Three people arrive and call the elevator, so three tasks are started to move it to the lobby. Because there is only one elevator, two tasks are queued until the first completes. The elevator arrives, the three people enter, and each selects their floor in the order 5, 3, 9, and a fourth person who has just arrived does not select any floor. The requests are processed in the order received, so the elevator proceeds to floor 5 as three more people request the elevator in the lobby just before the fourth person in the elevator presses the button for 10.

Because each button press causes a new task, the number of tasks waiting on the one elevator is growing with one task moving the elevator and six waiting for it to be available. Due to tasks being processed strictly in the order received, the fourth person in the elevator travels from the lobby to the floors 5, 3, and 9, returns to lobby and then finally reaches their floor. Meanwhile, someone repeatedly presses call on the eleventh floor as more people arrive in the lobby and press call resulting in requests for the two floors to become interleaved.

This example is intended to show that, without considering whether the task needs to be run or can be processed sooner, the number of tasks in the system can keep growing. Every task running in the system, whether actively running or suspended for resources, still uses resources such as memory. So naturally the more tasks that are present the more resources are required. As more resources are consumed by tasks waiting to move the elevator, tasks get delayed through lack of memory or because the system already has so many.

### ***Controlling the number and types of tasks in the system.***

CICS has a configurable limit on the maximum number of tasks that can run called `maxtasks`, which as mentioned in “Dynamically changing SIT values” on page 188 can be dynamically changed to meet changing requirements. However, to modify the value, a task needs to issue the command, but because the maximum number of tasks has been reached, no new tasks can be started to change the value. The same situation can exist when CICS becomes short of storage. Tools such as CPSM already have a task running to process the instruction issued through IBM CICS Explorer® or the CPSM Web User Interface (WUI). A far better position is to detect the build up of tasks sooner or prevent it occurring at all.

Aside from changing the entire elevator control application to detect duplicate requests and determine the best floor to travel to next, a quick fix might be to ensure that only one elevator task can be started at a time because there is a single elevator. CICS provides this capability for many releases through by allowing a transaction to be defined in a Transaction Class (`tranclass` or sometimes still called `tcclass`). While `maxtasks` controls the total number of tasks that can be in a region, `tranclass` enables a limit to be set on the maximum number of transactions in that class allowed to be active.

Originally, CICS only supported a transaction being in a **tc**class numbered 1 - 10 or “NO” to indicate that it was not in a **tc**class. The **tc**class was replaced with **tranc**class in CICS/ESA Version 4 Release 1, allowing any name up to eight characters in length to be used. By default, a transaction is in tranclass **DFHTCL00**, which is the original TCLASS(NO) and supplies definitions for tranclasses **DFHTCL01** to **DFHTCL10** to map to the original **tc**class 1 - 10.

A tranclass has two configurable values:

- **maxactive** controls the maximum number of tasks in the class that can be running with a valid range of zero to 999.

**Note:** Tranclass was not updated in CICS Transaction Server Version 5 Release 1 to match the new maxtasks value of 2000.

- **purgethresh** with a range of 1 to 1,000,000 or “NO” specifies the threshold that causes tasks to be purged. A value of “NO” means no tasks are to be allowed to queue.

Returning to the elevator example, the only one task can ever be active to move it, so defining a tranclass called “LIFT” with maxactive(1) is applicable.

The current tranclass of a transaction can be examined using the tranclass option of **exec cics inquire system** and dynamically changed to a defined tranclass using **exec cics set transaction**.

Similarly, the current values of **maxactive** and **purgethresh** can be examined and changed using **exec cics inquire / set tranclass** while inquire also supplies the number of currently active and queued tasks through options **active** and **queued**.

Similar to policies mentioned in “Tasks using large amounts of storage: Policy Support” on page 201, *CICS System Events* introduced in CICS Transaction Server Version 4 Release 2 enables the CICS region to be monitored for thresholds being reached. In particular, the system can be monitored for the number of tasks in the system reaching various percentages **maxtasks** for the system as a whole or within an individual **tranc**class.

## Long running transactions holding activity keypoints

As CICS runs, it periodically takes activity keypoints, the frequency of which are governed by SIT parameter **AKPFREQ**, recording in the CICS System Logstream DFHLOG that it has been taken. During the restart processing, CICS reads from the last entry in the logstream until an activity key point is found to determine all the uncommitted changes now in need of recovery. This means any information in the logstream before an activity key point is not required and can be deleted by CICS.

However, on occasions CICS might be unable to take an activity keypoint or trim the log, which it will report using message DFHLG0760 as shown in Example 18-5.

### Example 18-5 Example of DFHLG0760 message

---

```
DFHLG0760 03/13/2012 22:09:33 CIJ2GT Log stream SETUP.CICSGTJ1.DFHLOG not trimmed
by keypoint processing. Number of keypoints since last trim occurred: 17,968.
History point held by transaction: PST0, task number: 74748.
```

---

An occasional message like this will not severely affect the running CICS system, but it might be worth investigating whether the task in question can be enhanced. If this condition persists for a long time, there is a risk the CICS system log will become full, causing the region to fail with a long restart period because many records must be read as part of the restart. In

extreme cases, there might still be no log space available so having read all the records, the region fails again with the only option to initial start the region, losing all recovery information.

The DFHLG0760 can be processed by routing the message to an internal transient data queue (Intra-TDQ) for processing by a CICS task or to the system console to be processed by a tool such as IBM Tivoli NetView® for z/OS. If this is taking place on CICS Transaction Server Version 4 Release 1 or higher, the CICS task processing the message can issue a CICS event using **exec cics signal event** to issue alerts or take corrective action. CICS Transaction Server Version 5 Release 1 extended System Events to enable events to be emitted through CICS issued messages.

## Temporary Storage becoming full

Temporary Storage is provided for CICS transactions to share data storing multiple items of data in a queue identified by a 16 character name. Queues can be one of four types:

- ▶ Remote queues are those stored in a different CICS region.
- ▶ Shared queues are stored in a Coupling Facility Structure and accessed through a server (one per image and structure in use).
- ▶ Local auxiliary queues are stored by the CICS region in the DFHTEMP file.
- ▶ Local main queues are stored by the CICS region in memory.

As they are held in memory, main TSQs are lost when a region is restarted, whereas auxiliary TSQs are retained over a CICS restart until explicitly deleted. Unless the region is started with SIT parameter START set to INITIAL or COLD or TS=COLD. By default, commands to write to a TSQ do not have the option, causing them to suspend if there is insufficient space to complete the command. Unlike most CICS non-normal responses, the NOSPACE issued in this situation does not cause the application to abend. Unless it is checked for, *applications are not be aware* the write has failed.

If the TS space filling up is main storage, this can also affect other tasks in the region. Before CICS Transaction Server Version 4 Release 2, when it was moved to 64 bit storage, this was in 31 bit storage with no restriction on the amount of storage that can be used. TSMMAINLIMIT was also introduced in CICS Transaction Server Version 4 Release 2.

Since CICS/TS 1.SW1 or 1.2, **exec cics inquire tsqueue** has had option **lastusedint**, which indicates in seconds when the queue was last accessed. These data enable a user application to delete any queues determined to be no longer required. CICS Transaction Server Version 4 Release 2 introduced an extra parameter, **expiryint**, on the temporary storage model (**tsmodel**) that reflects through to a CICS hosted TSQ details when created specifying an unused threshold in hours when CICS is to automatically delete the queue provided it is not a recoverable or shared queue.

**Note:** **exec cics inquire tsqueue** can only be used on Local and Shared TSQs

As mentioned in “High space usage of Main Temporary Storage Queues” on page 199, the **tmainlimit** and **tmaininuse** options of **exec cics inquire tempstorage** can be used to determine the limit and current usage of main TSQ memory from releases **CICS Transaction Server Version 4 Release 2**. In addition to user written applications to monitor temporary storage usage, CICS provides the following facilities that can be used to detect developing problems in temporary storage.

CICS Application events introduced in CICS Transaction Server Version 4 Release 1 can be triggered by an **exec cics writeqts** command. These triggered events can be used checking for any execution failures, in particular any **NOSPACE** conditions.

CICS System Events extensions CICS Transaction Server Version 5 Release 1 enable events to be triggered when specified CICS messages are issued, such as these examples. Any messages issued to the console can also be intercepted and acted on by tools such as IBM Tivoli NetView for z/OS.

DFHTS1311 CII2GT Temporary storage data set is full and cannot be extended  
DFHTS1601 CII2GT Main temporary storage usage has reached xx% of TSMMAINLIMIT storage.

DFHTS1602 CII2GT Main temporary storage has attempted to exceed the TSMMAINLIMIT storage limit.

DFHTS1604 CII2GT Main temporary storage usage has fallen below 70% of TSMMAINLIMIT.

## DB2 Thread Use

Before CICS Transaction Server Version 4 Release 2, CICS reused threads into DB2 without recycling. This configuration sometimes caused issues in DB2 as they became “bloated” with DB2 resources. Storage in particular was allocated to the thread and not released. The **reuse1imit** parameter added to the **db2conn** resource specifies how many times any **db2thread** is reused before being terminated. The default value is set to one thousand to have negligible effect on CICS performance and reduce the likelihood of causing DB2 resource issues.

The minimum value of the existing **db2conn** parameter **purgecycle**, which specifies how long a DB2 protected thread remains unused before being reused or terminated, was also reduced from 30 seconds to five seconds with the default remaining at 30 seconds.

Both these values are visible and can be dynamically changed by using **exec cics inquire db2conn** and **exec cics set db2conn**.

## Identifying which programs are issuing commands related to issues

Having identified which commands might be causing issues within an environment, the next stage is determining which programs and applications are issuing those commands. In some environments, this is well defined with everything fully documented, enabling the information to be easily found. However, for most environments, documentation is not so finely detailed, emergency and other changes have been made and never documented, the application is from a third party so the internals are unknown, and code is older and less well defined. In these circumstances, you must employ other techniques:

- ▶ Scan source code
- ▶ Scan Load Libraries
- ▶ Examine CICS Trace
- ▶ Use CICS Exits
- ▶ CICS events
- ▶ CICS Interdependency Analyzer

### *Scan source code*

Scanning the source code is reasonably simple for small amounts of source code. However, it is complicated by having to allow for the CICS language translators accepting commands over multiple lines, allowing options to default, and accepting older options. For example, on **exec cics writeq ts qname(16 char name)**, **qname** has an older eight character version called **queue**, **ts** that can be omitted, but searching for just **exec cics writeq** might find **exec cics writeq td**.

This task becomes much more complicated for environments with large numbers of programs, large numbers of source libraries, and old programs that are no longer executed. Source repositories often have tools to scan the source looking for patterns rather than explicit strings to make this task easier.

## Scan Load Libraries

Scanning load libraries rather than source is, in some ways, easier because the CICS commands have a structured form, so identifying which libraries a program *can* be run from can easily be identified. Use of **exec cics inquire program** with the **librarydsn** option or the **CLDM** transaction (CICS Transaction Server Version 3 Release 2 and later) can be used to further identify libraries and programs of interest.

CICS supplies a utility **DFHEISUP** (CICS Transaction Server Version 1 Release 3 and later) that can scan load libraries looking for CICS commands that can be filtered to specific commands by input parameters. For more information about using **DFHEISUP**, see *CICS Transaction Server for z/OS Version 5 Release 1 Operations and Utilities Guide*, SC34-2863.

## Examine CICS Trace

When running commands, CICS records information to the CICS trace that is held in memory (internal trace) and written to the data sets allocated by the **DFHAUXT** and **DFHBUXT** DD entries (auxiliary trace) if active. Because CICS tracing, particularly auxiliary, can represent quite an processor burden on the region, many environments have at least auxiliary and potentially internal trace off.

**Note:** Exception trace entries are always written to the internal trace to aid in potential diagnosis of error conditions.

If active, these records can be formatted as part of a system memory dump for internal trace entries or using the **DFHTUxxx** utility for auxiliary trace records. The xxx represents the internal CICS release that generated the trace records. For instance, CICS Transaction Server Version 5 Release 1 is 680.

There are various trace entries that can be of use, but probably the most useful entry is **AP00E1**, which shows commands about to be run. Depending on the level of tracing active for the Application (AP) Domain at the time, these trace entries can show just the main CICS command being run such as **exec cics getmain** or all the appropriate options as well. While this method is usable, the ap00e1 trace records only show the CICS task number that has requested a particular command, not the program or transaction. To determine this information, previous trace records for the same task must be examined to find execution of **exec cics link**, **exec cics xctl**, or transaction initiation information. For more information about using CICS trace, see the *CICS Transaction Server for z/OS Version 5 Release 1 Problem Determination Guide*, SC34-2865

## Use CICS Exits

CICS provides many exit points to enable specialized customization of CICS activities:

- ▶ Changing where messages issued by CICS are sent (if at all) using exit point **XMEOUT**, for example to cause DFHLG0760 mentioned in “Long running transactions holding activity keypoints” on page 202 to be displayed on the MVS system console for detection by Tivoli NetView for z/OS.
- ▶ Override values specified on a command such as the name of a CICS file that has been changed but the running program cannot be updated for some reason.

**Note:** An exit can change a specified value, but it cannot change what options are specified or the command to be run by CICS.

- ▶ Tools such as CICS VSAM Transparency intercept normal application program File Control requests, issue their own call to DB2, suppress the original commands from being

issued, and set the appropriate CICS return code so the application is unaware its request has been routed to DB2 instead.

The more an exit is started by CICS, the bigger the potential processor burden it becomes on the running system. Care must be taken to ensure that the exit program code is threadsafe and complies with CICS requirements for an exit at the particular point. It is an implementation choice which exit points are used. The following are examples of two exit points:

- ▶ XEIIIN, which is started before the execution of any exec cics command
- ▶ XTSEREQ, which is started before the execution of a temporary storage API request

Although there are many exit points available with different information provided to the started exit program depending on what that point is for, many can issue exec CICS commands or CICS exit programming interface (XPI) calls. An exit program using the XPI at exit point `xei in` can perform these tasks:

1. On invocation, the exit is passed various data including the address of the CICS command parameter list, which contains a pointer to the EXEC interface descriptor (EID).

The first two bytes of the EID give the function code of the command about to run. A list of these values is available in the *CICS Transaction Server for z/OS Version 5 Release 1 Application Programming Reference*, SC34-2845 under “EIBFN”.

2. A crude implementation might use the XPI call `SYSTEM_DUMP` or `TRANSACTION_DUMP` to trigger a system or transaction memory dump if the command is one of interest. However, a less disruptive implementation would XPI call `INQUIRE_CURRENT_PROGRAM`, which can supply the running program name, the library it was loaded from, and which program started it if required. XEIIIN is also supplied with the current program name as an input parameter.

This information can then be written out to an MVS logstream using the XPI call `WRITE_JOURNAL_DATA`.

**Important:** If an exit program does not meet all the requirements specified in the documentation, it can have a detrimental impact on the region. Problems caused can include unpredictable behavior, performance issues, and instability that can cause effects outside the region.

For more information about writing CICS exits, see the *CICS Transaction Server for z/OS Version 5 Release 1 Customization Guide*, SC34-2847.

### **CICS events**

Initially added in CICS Transaction Server Version 4 Release 1 and expanded in subsequent releases, events enable notifications to be generated when certain criteria are met. Events are split into two types:

- ▶ Application events are based on a certain command being issued, for example an `exec cics link` with the ability to filter the specification further based on information such as the program being linked to, contents of the command or transaction being run. An application can also explicitly request that an event is generated by using `exec cics signal event`. Except for explicitly requesting that an event be generated, application events can be used with no change to the application programs themselves. A generated application event can contain various data that are specified when defined including details such as the transaction running and the currently running program. In the various scenarios mentioned above, CICS Transaction Server Version 5 Release 1 can only capture temporary storage and program link activity, but various other commands are also supported.



- System events have been mentioned in previous sections and enable notifications to be generated based on thresholds in the system itself such as percentage of active tasks in a particular transaction class or the region's maximum in "Controlling the number and types of tasks in the system." on page 201 and CICS messages in "High space usage of Main Temporary Storage Queues" on page 199

In addition to the CICS supplied facilities to place generated events on Message Queue Queues, send over http(s), write to a TSQ, or start a new transaction, a sample is provided showing how to generate customized versions. For more information about using CICS events, see the CICS Information Center, various Redbooks and CICS Support Pack CA1Y "Send email from CICS Transaction Server for z/OS" that demonstrate sending emails using CICS events at:

<http://www.ibm.com/support/docview.wss?uid=swg24033197>

### ***CICS Interdependency Analyzer***

The CICS Interdependency Analyzer (IBM CICS IA®) documentation describes CICS IA as a CICS runtime tool with three primary purposes:

- To identify interdependencies of transactions
- To identify transaction affinities
- To identify and analyze resource usage flow

Data that are captured by CICS IA are initially held in VSAM files or an MVS logstream with batch jobs supplied to upload the data to DB2. DB2 is the more flexible format to use when working with the captured data. Supplied sample batch jobs, CICS transactions, and use of the CICS IA plug-in for CICS Explorer enable the data to be analyzed to determine which programs are issuing commands of interest (including those to DB2, WebSphere MQ, and IMS).

CICS IA also provides a load module scanner. In addition to the CICS commands present in load modules, it also reports on DB2, WebSphere MQ, and IMS commands.

For more information about CICS IA, see *CICS Interdependency Analyzer for z/OS Version 5 Release 1 User's Guide and Reference*, SC34-2811.

## **18.4 Reducing shutdowns through runtime dependencies**

Many CICS regions have a dependency on resource managers outside of CICS, without which some or all of the applications running on the region are unable to function. In some circumstances, a resource being unavailable means the region cannot function at all, causing it to terminate. For example, loss of MVS logger means that CICS is unable to log any recoverable activity, causing it to terminate abnormally.

Whether a region should be terminated because a resource has been detected as unavailable or is about to become so is a value judgment. Make this judgment based on a resource by resource (or application by application) basis because in some instances unavailability of a less important facility might be less of an issue than terminating the entire region.

The following list of questions in no specific order might be of assistance in reducing the perceived unavailability from a user's perspective, although the expense of implementing some solutions, particularly without a Parallel Sysplex environment, might be impractical:

- Is an unavailable response acceptable?

For example, if DB2 is unavailable, the application code can detect this so rather than failing, it returns a response to the caller. While this might not be the ideal response, the calling application can then pass the same response to its caller or try another route.

- Can the request be queued within CICS?

Depending on the nature of the activity, performing the request later rather than failing to process the request might be suitable. For example, if an item is ordered on the internet but the order database is unavailable, informing the customer that their order has been queued for processing is a more acceptable response than asking them to try later and risk them going to a competitor.

- Can the request be rerouted by CICS?

The information so far as been written from the perspective of a single CICS region not using Parallel Sysplex, but it might be possible to pass requests to another CICS region to process the request. In a Parallel Sysplex environment, determining the region cannot process the request and not routing to it at all is a simpler process.

- Is there another instance of the external resource manager?

Multiple instances of DB2 and WebSphere MQ can be active on single image, but members of the same DB2 or WebSphere MQ group and CICS can be configured to connect to any member of the group. This means that other than a brief interruption, availability from a CICS perspective can be maintained by starting a second instance on the image and switching CICS to it. The appropriate commands are **exec cics set db2conn** (options **connectst**, **db2groupid** and **db2id**) and **exec cics set mqconn** (options **connectst** and **mqname**). If one of these connections is terminated with outstanding units of work, they might not be resolved until CICS reconnects with the original instance. For more information, see the documentation for the **resyncmember** option of both commands in the *CICS Transaction Server for z/OS Version 5 Release 1 System Programming Reference*, SC34-2872.

- Can the external dependency be made more robust or moved dynamically?

For example, CICS has three types of coupling facility (CF) servers that have a heavy dependency on CF structures. If a server has a problem accessing its particular structure, it will fail. If this is due to a problem with the CF or CF structure itself, all data in the structure is lost, causing an integrity issue. For environments with more than one CF, structures can be moved between CFs using the operator **setxcf** command to enable one to be shut down for maintenance. However, it is still vulnerable to unexpected issues affecting access to a CF structure. Making the structure a duplexed structure removes this vulnerability, as two versions of the structure are maintained.

- Can tasks be restarted more quickly?

Many environments use CICS File Owning Regions (FORs) to enable multiple distinct regions access to the same files, which being a single point of failure can become critical. Both CICS regions and Coupling Facility Servers can be automatically restarted by the MVS automatic restart manager (ARM) so if they do fail, they can be restarted much more quickly on a different image.

**Note:** If started on a different image and using TCP/IP Connections (IPIC), the TCP/IP environment needs to be able to handle the address used by region now being on a different image.

- Can single points of failure be removed?

As mentioned previously, an FOR is a single point of failure that, if unavailable for any reason, can cause problems for multiple regions and applications. Using VSAM record-level sharing (RLS) is a method of enabling each individual region to access the files directly rather than depend on another region. However, there are some considerations when switching to using RLS files, which are described in *IBM CICS Performance Series: CICS and VSAM RLS*, REDP-4905. An alternative to an FOR is CICS VSAM Transparency, described in 18.5, “Reducing shutdowns to allow resource access by non-CICS tasks” on page 209.

## 18.5 Reducing shutdowns to allow resource access by non-CICS tasks

The most likely resources a CICS region would prevent access to by non-CICS tasks are VSAM files, sequential files, and spool files. As described in 18.2.4, “Dynamic allocation of files and Extrapartition Transient Data Queues” on page 186, if the file is dynamically allocated, closing the file also deallocates the file from CICS. However, you need to consider the issues described in 18.4, “Reducing shutdowns through runtime dependencies” on page 207.

The following sections detail some of the reasons a resource might be required by processes outside CICS and solutions that might be appropriate.

### 18.5.1 Backups

Generally to take a backup of a file, it must be closed. Otherwise, what has been copied as the backup might not be a true representation of the actual file contents. For instance, data might still be in memory buffers of the task using the file. It might not be possible to even open the file to copy the data because of share options.

Backup-while-open (BWO) is a facility that enables DFSMSdss to take a backup of a VSAM data set even when applications have it open for update with full integrity of the copied data that are specified as required in one of two ways:

- Specifying **BWO(TYPECICS)** on the catalog entry for the data set. This parameter can have other values that include **TYPEIMS** and **TYPEOTHER**, but CICS regards all values other than **BWO(TYPECICS)** as indicating the data set is not eligible for BWO. The BWO setting on the catalog is used by CICS for any file open in RLS mode and for files opened in non-RLS mode if the SIT parameter **NONRLSRECOV=VSAMCAT** (the default) is specified.
- Specify **BACKUPTYPE(DYNAMIC)** on the CICS file definition for any files opened in non-RLS mode if SIT parameter **NONRLSRECOV=FILEDEF** is specified or the BWO setting is not defined in the catalog.

If the file being backed up with BWO also has a forward recovery log stream name specified, in addition to the backup taken at a specific point in time it can be recovered to any subsequent point in time by reapplying all updates subsequently made using a tool such as CICS VSAM Recovery for z/OS Version 5 Release 1. For more information about using BWO, see the CICS Recovery Restart Guide.

## 18.5.2 Batch access to data sets

There are various tools and techniques available for both VSAM and sequential files. Some can be used without change to programs and others require program changes to adopt. Which approach is the best choice varies between environments and possibly between individual programs based on factors including available tools and products, how easily programs can be changed and long-term strategies for the environment. Some of the tools available are listed in the following sections.

### Change to read only access non-RLS in CICS

Opening a file for read-only (RO) access in CICS and read/write (RW) in a batch program is possible provided that the file is defined with share option 3. As mentioned in 18.5.1, “Backups”, accessing a file as non-RLS from batch can affect the recoverability due to gaps in the forward recovery logstream data. However, remember that *data integrity in the RO access* is not assured because of data buffering and caching, particularly from multiple images or through alternative index entries. If this is not considered an issue, CICS can either close the files using **exec cics set file closed disabled**, change its access to RO using **exec cics set file notaddable notdeletable notupdatable**, and then reopen the file using **exec cics set file open**. If the file is accessed RW in RLS mode at the same time as switching to RO, it must be changed to non-RLS with the option **NOTRLS** because a file cannot be opened in RLS mode and RW non-RLS concurrently. Reverting to normal operation is achieved by undoing the changes.

This method causes an interruption to the file’s availability, which can be avoided by designing the applications to check for a flag of some type, for example the existence of a Shared TSQ or a particular value in the CICS Work Area (CWA). When they detect that the flag is set, the applications use a different CICS file. After a pause to allow any existing access to the RW files, they are closed and the batch can start.

A second method to avoid an interruption to the file availability is to have two versions of the applications in different load libraries, one using the RW files and one using the RO files. Each library is defined in a different dynamic program library (see 18.2.3, “Dynamic program library management” on page 186) and normally runs with the RW entry enabled and the RO disabled. Before running the batch, the RO program library is enabled and, depending on search priorities, the RW library is disabled using **exec cics set program phasein** (see 18.2.1, “Reloading programs” on page 184).

**Note:** If you choose to have the same data set open RW and RO in the same region using RLS for the RW file and non-RLS for the RO file, an exit program at exit point **XFCRLSCO** must be used as described in *CICS Transaction Server for z/OS Version 5 Release 1 Customization Guide*, SC34-2847.

### CICS Batch Application Control

In these scenarios above, a series of steps must be undertaken either manually or using various automation tools. However, either assumes that a set time batch jobs will be running or require some mechanism to determine when the batch processing has completed.

CICS Batch Application Control (CICS/BAC) enables batch programs to issue **exec cics set** commands against files, TDQs, transactions, and programs, start the CICS master terminal facility CEMT, start transactions, and link to CICS programs from a batch job. Set commands can be issued against groups of resources and lists of groups.

Using the second of the scenarios in “Change to read only access non-RLS in CICS” on page 210, a new first step of a batch job can issue a request to CICS/BAC to run these tasks:

- ▶ Link to a CICS program to set the flag indicating that the RO files are to be used, which also performs an **exec cics delay** to allow any existing tasks to complete.
- ▶ Issue a set file closed for the RW files.

Having delayed to allow for the files to close (or have JCL that causes the execution to wait for the files to be available or call a second cics program that ensures the files are closed), the batch job continues with its original steps to perform its original processing. After the original processing is complete, then runs a new final step to run these tasks:

- ▶ Issue a set file open for the RW files.
- ▶ Link to a CICS program to clear the flag indicating the RO files are to be used. This program might also have issued an **exec cics delay** to allow time for the files to open or be passed a list of files to check are open before clearing the flag in a command.

**Note:** CICS/BAC allows files to be accessed in RLS mode and issues an **exec cics set dsname** to quiesce and unquiesce the files in response to open and close requests. This might cause unexpected results as this causes all RLS mode files using the base VSAM file in all regions to close not just the specific file in the specific region.

Although this is often the wanted result, in the scenario above assume that multiple regions were being instructed in turn to set the “use RO files” flag and then close the file rather than the flag being in some shared location. As soon as the first file is set to closed on the first region, other regions using the file will be informed that the file is being quiesced and also close the file while applications are still using it. In this configuration, the entire BAC processing might be better processed by a single program linked to in each region that sets the flag and then closes the file. A later CICS/BAC instruction to close the file can still be issued to a single region to quiesce the file to ensure that no RLS access is attempted if wanted.

For more information about CICS/BAC, see *CICS Batch Application Control for z/OS Version 1 Release 1 Modification 1 User's Guide*, SC34-6321.

## Transactional VSAM

Transactional VSAM was added to z/OS 1.4 and CICS Transaction Server Version 1 Release 3 CICS Transaction Server Version 3 Release 2. It can be used to access files in RLS mode from batch programs. In addition to enabling the file to be used concurrently by CICS and batch programs, it improves the recoverability of files with a forward recovery log because all updates are now contained in the log. However, updates to a file in non-RLS mode by a batch program are not recorded in the forward recovery log unless performed by the program itself. Although a batch program can use Transactional VSAM with only changes to its JCL, it is performed as a single Unit of Work, potentially locking large numbers of records. Therefore, consider updating the program to perform *syncpointing*.

For an introduction to Transactional VSAM, see Technical Exchange Presentation CICS and Transactional VSAM at:

<http://www-01.ibm.com/support/docview.wss?uid=swg27009511&aid=1>

For more detailed information, see *DFSMSdfs Overview and Planning Guide*, SG24-6971 or the CICS Information Center.

## CICS VSAM Transparency

CICS VSAM Transparency (CICS VT) enables VSAM files to be moved into DB2 by intercepting the calls made to access the original VSAM files, processing the request in DB2, and returning the result to the caller as though the request had been performed against the VSAM file. In CICS, this is achieved by use of CICS exits that have already been mentioned in “Use CICS Exits” on page 205 and for batch jobs through JCL changes. To aid migration, CICS VT can be run using dual mode facility (DMF) on a file by file basis, causing both DB2 and the original VSAM file to be accessed and the results from both compared. If a difference is detected, the requesting application is abended to ensure integrity and, depending on the configuration of the DMF, various diagnostic information is captured. The main purpose of DMF is ensure correct mapping of data in CICS/VT from a VSAM record to a row of DB2 data with multiple columns.

In addition to making the data available concurrently in batch and CICS, CICS VT enables new or updated applications to access the data directly in DB2. It also takes advantage of the relational capabilities of Structured Query Language (SQL).

For more information about CICS VT, see the *CICS VSAM Transparency for z/OS Version 2 Release 1 User's Guide*, SC34-7249.

CICS/VT 2.1 does not support these items

- ▶ Direct access to a VSAM alternate index cluster as though it had been defined as a VSAM base cluster.
- ▶ Entry-sequenced data set (ESDS) data set types.
- ▶ Linear data sets.
- ▶ Processing using relative byte address (RBA) access.

## CICS Modern Batch Feature Pack

Available for CICS Transaction Server Version 4 Release 2 and later, this CICS feature pack (Program Number 5655-Y50, FMID HCIF51B) enables batch applications to be run inside CICS Java virtual machine (JVM) server in parallel with existing CICS applications. However, unlike a traditional CICS application running exclusively inside the CICS environment, because these tasks are still run as batch tasks, all the batch scheduling facilities can be used. Additionally, these tasks can use any existing CICS programs and, because they are running inside the CICS JVM, use CICS Java application programming interface (JCICS API) calls to access CICS resources directly.

Because this environment is based on Java, it is probably more suited to newer batch applications than older ones traditionally written in other languages.

For more information about this facility, see the CICS Information Center.

## WebSphere XD Compute Grid Support Pack CN11

This is an alternative to the CICS Modern Batch Feature. The main difference is that, while available for CICS Transaction Server Version 4 Release 1 and later, this is supplied on an “as is” basis with less capability. The download page for this support pack at <http://www-01.ibm.com/support/docview.wss?uid=swg24027213> carries the statement that clients should use the CICS Modern Batch Feature Pack unless using CICS Transaction Server Version 4 Release 1. For more information about using this, see *New Ways of Running Batch Applications on z/OS: Volume 1 CICS Transaction Server*, SG24-7991.

## SPIN option on spool output files

Two reasons spool output files can cause a requirement to recycle a CICS region are the amount of output that is produced and the need for it to be processed by some external process. These spool data sets cannot be processed until they are closed, so the JESMSGLG and JESYSMSG data sets in particular can become very large and take significant amounts of time to process.

The SPIN=UNALLOC parameter introduced in z/OS 1.13 added the ability for spool data sets to be split off without interrupting the job to be processed and purged, reducing the amount of spool in use. The splitting can automatically occur using specified criteria such as amount of data, time of day or time since opened, as the result of an operator command or the job closing the output file. For more information, see the information provided in “z/OS components” on page 21.

## Rerouting or suppressing

Most CICS output to spool is either generated as CICS messages or as information written to Extra TDQs. It can be rerouted or suppressed if wanted using CICS Exits that have already been mentioned in this chapter. For example, an unwanted CICS message or write to an Extra TDQ can be suppressed completely or rerouted to an Intra TDQ for processing by a CICS task.

Many CICS messages are written to various different TDQs that, when using the CICS supplied TDQ definitions, are defined as indirect TDQs routed to the CSSL Extra TDQ using output DD card MSGUSR. This process is used because each individual TDQ is used for messages from different areas of CICS such as security or terminal control. By using different queues, you have the flexibility of varying the processing of messages from different areas. If you want to dynamically change the installed definition for a TDQ, remember that CICS will not allow a TDQ whose four character name starts with the letter “C” to be discarded. However, a new definition can be installed to replace an existing closed and disabled queue provided the type of queue is not changed. Because the supplied definitions are generally for indirect queues, to achieve this without restarting CICS, the definition can be replaced to route the queue to a new queue that has the required attributes.

If you want to discard all messages that are written to a particular TDQ, you can define the target queue as an Output Extra TDQ defined with **DSNAME(DUMMY)** or specifying **DD DUMMY** in the JCL.

**Note:** **DSNAME(DUMMY)** permits the definition to be dynamically changed if desired.

This is preferable to disabling the TDQ because the generator of the TDQ data would receive a notification that the write to a disabled queue has failed, but will be unaware that the output is being discarded by **DSNAME(DUMMY)** or **DD DUMMY**.

## Use of Generational Data Group output data sets

As mentioned previously in 18.2.4, “Dynamic allocation of files and Extrapartition Transient Data Queues” on page 186, files used by CICS can be dynamically changed using the CICS Programming Interface. Therefore, a process can be used where an output Extra TDQ defined with a disposition of **NEW** is closed and disabled, the data set name changed to a new name, and the data set enabled and opened.

The name that is used must be generated programmatically, perhaps using the date and time the change has occurred. Another option is for the programs to use a generation data group (GDG). The programmer should choose which option is most suitable for the application.

## 18.6 CICS still must be stopped, so what next?

When it is necessary to stop CICS, you can start a second CICS to take over inbound work. Using generic names in VTAM, RLS, port sharing, and dynamic VIPA can make the changeover transparent to the user. CPSM can make the management of the interface much easier, particularly through Workload Manager.





## DB2 considerations

This chapter describes ways to manage planned outages in a DB2 for z/OS environment. Because DB2 is a database manager, there are two aspects of availability to be considered:

- ▶ The availability of the DB2 subsystem. The primary audience that will be interested in this information is DB2 system programmers. This chapter covers enhancements in DB2 that provide you with more flexibility regarding when the DB2 subsystem will be restarted.
- ▶ The availability of the data managed by DB2. Significant enhancements have been made to DB2 to allow you to change the data infrastructure with little or no loss of access to the data. The primary audience for this type of information is DB2 database administrators (DBAs).

Because of the two audiences for this chapter, it is broken into two sections: one section contains information that is primarily of interest to DB2 system programmers, and the other section is aimed at DB2 DBAs.

This chapter includes the following sections:

- ▶ DB2 subsystem or data sharing group member outage
- ▶ Database changes

## 19.1 DB2 subsystem or data sharing group member outage

Access to DB2-managed data requires that at least one of the DB2 subsystems associated with that data is available. Even though there have been many enhancements to DB2 to reduce the number of situations that require the DB2 subsystem to be restarted, some changes still require a DB2 restart. Therefore, if you have a near-continuous availability requirement, you that you must implement DB2 sysplex data sharing. This allows you to recycle one DB2 subsystem while maintaining access to the DB2 data through another member of the data sharing group.

**Note:** This chapter assumes that there are no affinities that restrict the application to a single system. If such affinities do exist, this chapter does not address their elimination because that information is provided in other IBM documents.

This section focuses on enhancements to DB2 that enable you to change DB2 without restarting it. It also describes the considerations related to managing the application of service to DB2 and migration to new releases.

Table 19-1 lists many of the reasons for recycling a data sharing member and where these are described.

*Table 19-1 Reasons for recycling a data sharing member*

Reason for recycle	Applies to	described in
IBM DB2 product maintenance	All versions	19.1.1, “Applying DB2 service (PTF/RSU/CST)” on page 216
Migration to new version of DB2 Includes fallback to a prior version or state (CM*, ENFM* NFM*)	All versions	19.1.2, “Version to version upgrade/fallback” on page 226
Add an Active Log	Versions before DB2 10	19.1.5, “Dynamic addition of active logs” on page 228
Pick up <i>some</i> DSNZPARM changes	Selected parameters since DB2 Version 8	19.1.3, “DSNZPARM changes” on page 227
Convert BSDS to Extended RBA/LRSN Format	DB2 V11 (optionally)	19.1.4, “Convert BSDS to Extended RBA/LRSN in DB2 11” on page 227

One of the events that will always require a restart of DB2 is the application of service to DB2. Therefore, this section starts with a description of the considerations relating to how you manage your DB2 environment to provide the flexibility to quickly and easily move back and forth from one service level to another.

### 19.1.1 Applying DB2 service (PTF/RSU/CST)

The best practice for data sharing is for rolling maintenance, where the outage during which a member is recycled for new service is concealed because the workload continues to run on the other DB2 members. Whatever strategy you have, keep these goals in mind:

- You want the flexibility to change *some* DB2s in a system (that is, not all DB2s in a system should point at the same DB2 data sets)

- ▶ You want the flexibility to change *some* DB2s in a data sharing group (that is, you do not want to upgrade the whole data sharing group at the same time).
- ▶ You want the flexibility to have more than one DB2 subsystem per data sharing group per z/OS system (meaning that you effectively must use DB2 group attach to get the full benefit).
- ▶ You want the flexibility to have one set of batch or utility JCL in a PROCLIB and to be able to run that JCL from any system on which they are submitted.
- ▶ One of the things to be considered in relation to this is which DB2 member a job attaches to when multiple members of the data sharing group are active on the same system. This is a more complex topic than you might think. For a description of this topic, see the section titled “How DB2 chooses a subsystem name” in *DB2 for z/OS Installation and Migration Guide*. This topic is a consideration outside of maintenance.

The following approach to maintenance is general and is designed to provide flexibility. This example is based on experiences in the Washington Systems Center with a sysplex called WSCPLEX. Every installation has its own standards.

**Important:** Avoid LINKLIST “lock in”. Placing product libraries in the LINKLIST is easy, which is why most users have done it. However, it means that all DB2 subsystems on an LPAR are upgraded to a new service level or version at the same time. That affects not only test and development, but also production DB2. The key to best practice maintenance is the use of STEPLIBs.

Use STEPLIBs to allow for rolling maintenance and situations where there is more than one member of the group on the same system or multiple members of different groups on the same system. Parts of this section are illustrated from experience with the WSCPLEX. The example has one TSO logon procedure, but uses different DB2 logon commands to concatenate the set of DB2 libraries in front of those for ISPF. The only library in the LINKLIST is SDSNLINK. Placing other libraries in LINKLIST, as many clients have done in the past, is easier, but provides less flexibility for rolling maintenance, so that was not done.

The following libraries and the DB2 UNIX file system are maintained together:

- ▶ SDSNLOAD: PDSE data set that contains DB2 program code
- ▶ SDSNLOAD2: PDSE data set that contains JDBC and SQLJ DLLs.
- ▶ DB2.ZFS: The z/OS UNIX file system that contains the Java portion of the IBM Data Serving Driver for JDBC and SQLJ for type 2 z/OS connectivity. It also contains the ODBC code.

The example contains three sets of libraries that are used across the members of the data sharing group. SDSNLOAD is used in the example, but in reality the set of libraries also includes SDSNLOAD2 and DB2.ZFS:

- ▶ *Current service*.SDSNLOAD, which can be RSU1404
- ▶ *New service*.SDSNLOAD, which can be RSU1407
- ▶ *Previous service*.SDSNLOAD, which can be RSU1401

The following libraries are slightly different:

- ▶ SDSNEXIT: A user library that contains DSNZPARMs, DSNHDECP, and user exits. It is concatenated in STEPLIB, in front of SDSNLOAD. It is version specific, but not normally service level specific.
- ▶ SDSNLINK: Contains the DB2 Early code for subsystem initialization. It does not appear in STEPLIB because these modules must be placed in the z/OS LINKLIST.

All the libraries must be APF-authorized: SDSNLOAD, SDSNLOAD2, DB2.ZFS, SDSNEXIT, and SDSNLINK.

**Tip:** You most likely want more “generic” names for SDSNLOAD, SDSNLOAD2, and DB2.ZFS, so that you do not have to authorize specific libraries. The example uses actual RSUxxxx as qualifiers because it is easier to show the actual service levels.

## Cataloged procedures and ICF catalog alias

The cataloged procedures in PROCLIB do not change when maintenance is applied. This is accomplished by the implementation of an ICF catalog alias. Each DB2 member has an alias that can be dropped and redefined to point to the required load library. See Example 19-1.

*Example 19-1 Sample IDCAMS statement to define alias for SDSNLOAD library*

---

```
DEFINE ALIAS -  
    (NAME(<hlq>.<group_att_name>.<member_name>.<SDSNLOAD>) -  
    RELATE(<hlq>.<group_att_name>.<service_level>.<SDSNLOAD>)) -  
    CATALOG(<user_cat>)
```

The DB2 STEPLIB would have <hlq>.<group\_att\_name>.<member\_name>.<SDSNLOAD>

---

The ALIAS points to the name of the actual data set, *hlq1.groupattachname.servicelevel.SDSNLOAD*. The library name that is referenced in SYS1.PROCLIB for the started tasks (MSTR, DBM1, IRLM, DIST) is represented as *hlq1.groupattachname.member.SDSNLOAD*, where:

- ▶ *hlq1* is the high-level qualifier for the data sets
- ▶ *groupattachname* is the 4-position DB2 group attach name
- ▶ *member* is the 4-position subsystem name (SSID)
- ▶ *servicelevel* is your designation for a particular service level

As an example, assume a 4-way data sharing group, WSCDBP0, with members DBP1, DBP2, DBP3, and DBP4. The group attachment name is DBP0. Assume that the current service level is RSU1404.

If the library name referenced on the STEPLIB in the DBP1 procedure is DB2HLQ1.DBP0.DBP1.SDSNLOAD, the actual data set used would be DB2HLQ1.DBP0.RSU1404.SDSNLOAD, where RSU1404 designates the current service level. The IDCAMS statements to define the alias are shown in Example 19-2.

*Example 19-2 Relating STEPLIB reference for started tasks to actual library name*

---

```
DEFINE ALIAS NAME (DB2HLQ1.DBP0.DBP1.SDSNLOAD) -  
RELATE (DB2HLQ1.DBP0.RSU1404.SDSNLOAD)) -  
CATALOG (ICFCAT.DBP0.DBP1)
```

---

Each of the DB2 subsystems has a similar set of aliases, with the second to last qualifier identifying the subsystem. At the moment, the aliases for all four DB2s point at the same actual library: DB2HLQ1.DBP0.RSU1404.SDSNLOAD.

All the cataloged, JCL, and TSO logon procedures have STEPLIBs so that new maintenance or release levels can be introduced by individual members when multiple DB2 subsystems from the same or different data sharing groups are running on the same system.

The STEPLIB for batch jobs that use WSCDBP0 is simpler because it does not have a member name associated with it. Its alias is defined as shown in Example 19-3.

*Example 19-3 Relating STEPLIB reference for batch and utilities to actual library name*

---

```
DEFINE ALIAS NAME (DB2HLQ1.DBP0.SDSNLOAD) -  
RELATE (DB2HLQ1.DBP0.RSU1404.SDSNLOAD)) -  
CATALOG (ICFCAT.DB2HLQ)
```

---

Aliases are also defined for SDSNLOD2.

The group attach name that is specified on a subsystem parameter in a batch job allows it to connect to any member in the data sharing group. This name is helpful when multiple DB2s are active on the same system. It allows a job to be submitted to run on any system that has a member of that data sharing group without having to specify the actual subsystem name. The group attach name can also be used for the subsystem parameter for utilities. For more information, see Chapter 2. “Preparing your system to install or migrate DB2” in *DB2 for z/OS Installation and Migration Guide*.

**Important:** The attach code and utilities found in the batch procedures are designed to operate with an upwards/downward level of DB2 code within a specific version. In other words, the DB2 code that is loaded into the application's address space (DSNUTILB for utilities, CAF, RRS, TSO) can be at a different maintenance level than DB2 is running.

Guidance on running utilities, CALL, and TSO attach in a mixed release (coexistence mode) environment is provided in *DB2 for z/OS Installation and Migration Guide*.

## SDSNLOD2 and the IBM Data Server Driver for JDBC and SQLJ

Your maintenance process should consider the relationships between SDSNLOAD, SDSNLOD2, and the DB2 UNIX file systems. The general rule is that everything is expected to be at the same level. There is a tight relationship between the SDSNLOD2 and UNIX file system code *within* an application address space.

**Note:** The DB2 UNIX System Services files can be stored in either an HFS or a zFS. However, because HFS has been functionally stabilized, and zFS is the strategic file system, zFS is used for the remainder of this chapter.

The driver/application type code is loaded from SDSNLOD2 and the zFS has a dependency on SDSNLOAD (and also SDSNEXIT). This dependency is generally *not* sensitive to the maintenance of SDSNLOAD. In rare cases, there might be some specific minimum requested level as an SDSNLOAD prerequisite APAR/PTF pointed out by a ++HOLD.

For IBM Data Server Driver for JDBC and SQLJ, for type 2 connectivity for z/OS (T2z), the Java portion of the driver is loaded from the file system, while the native DLL portion of the driver loads from the SDSNLOD2 data set. For STEPLIB concatenation, it does not matter whether SDSNLOAD or SDSNLOD2 are concatenated behind SDSNEXIT, as the modules distributed by IBM are different.

The requirement is that the SDSNLOD2 definition (ICF alias or STEPLIB) for each DB2 member in a data sharing group must correspond with that of DB2.ZFS on the file system. This can be done by defining a symlink within zFS, dynamically switched in sync with the ICF alias definition. A general description is found in “File system activities” on page 223.

## SDSNLINK

SDSNLINK contains modules that you must place in the link list because they are loaded at subsystem initialization during IPL. For Version 10, the load module library SDSNLINK contains modules that are called early (ERLY) code. The general statement is that early code is upward/downward compatible by one version. If there is a “skip” release, such as occurred in DB2 Version 8, then early code is compatible with the earlier version. These are the levels at the time this book was written:

- ▶ Version 8 early code with the required maintenance is compatible with Version 10.
- ▶ Version 9 early code is compatible with Version 8 and Version 10.
- ▶ The Version 10 early code is compatible with Version 8 and Version 9.
- ▶ The Version 11 early code is compatible with Version 10.

If you are upgrading, maintenance to early code or an installation of new early code requires that you activate the early code, which can be done in a couple of ways. The better option is to issue the DB2 command **-REFRESH DB2,EARLY**. If you use this command, the subsystem that is having its early code refreshed must be stopped when the command is issued. To initialize a new DB2 subsystem, issue the **SETSSI z/OS** command, which runs the early code initialization routine and updates the subsystem vector table (SSVT). Another option to activate the code is to IPL z/OS. An IPL is required if Version 8 early code is in use for that subsystem.

SDSNLINK has these properties:

- ▶ Contains early code
- ▶ Can be shared by multiple subsystems and versions of DB2
- ▶ Is APF-authorized

Because the early code can be shared among the subsystems and versions of DB2, you can copy the current service level of SDSNLINK modules into a library, generically named and APF-authorized, that is in LINKLIST. There is little maintenance during the life of a DB2 version to the early code and, if it became necessary, there are instructions in the ++HOLDDATA.

**Note:** The -REFRESH command works if you copy the individual modules into an existing LINKLIST library. If you replace or add a library to the LINKLIST, an IPL is required.

## A maintenance scenario

The example has three sets of data sets for different service levels (previous, current, new) for the WSCDBP0 data sharing group. These were all copied from the SMP/E target libraries. The following are the three service levels for SDSNLOAD:

- ▶ DB2HLQ1.DBP0.RSU1404.SDSNLOAD is the current service level
- ▶ DB2HLQ1.DBP0.RSU1407.SDSNLOAD is the new service level
- ▶ DB2HLQ1.DBP0.RSU1401.SDSNLOAD is the previous service level

Example 19-4 showed how to define the alias for DBP1 to one of these data sets. If you want to move DBP1 to service level RSU1407, you would shut that DB2 down, delete the current aliases, then redefine the aliases to point to the RSU1407 level of the data sets as shown in Example 19-4.

*Example 19-4 Switching the started task Alias to point to new service level*

```
DELETE -
    DB2HLQ1.DBP0.DBP1.SDSNLOAD -
    ALIAS -
    CATALOG (ICFCAT.DBP1)
```

```
DEFINE ALIAS -  
    (NAME(DB2HLQ1.DBP0.DBP1.SDSNLOAD)      -  
     RELATE(DB2HLQ1.DBP0.RSU1407.SDSNLOAD)) -  
    CATALOG (ICFCAT.DB2HLQ)
```

---

When you restart DBP1, it comes up pointing at the new service level data sets. The other members remain at the older service level, as does batch.

Rolling through maintenance is a matter of performing the same tasks for every member at time that you want.

If errors arise in the maintenance, reverting to the previous service level is as simple as stopping DB2, changing the aliases to point back to the previous set of data sets, and restarting DB2. The symlink for the file system would also be deleted and redefined to point to the same service level.

This scenario achieves the first three goals that are listed at the beginning of this section:

1. Flexibility to change *some* DB2s in a system (that is, not all DB2s in that system should point at the same DB2 data sets)
2. Flexibility to change *some* DB2s in a data sharing group (that is, you do not want to upgrade the whole data sharing group at the same time).
3. Flexibility to have more than one DB2 subsystem per data sharing group per z/OS system (meaning that you effectively must use DB2 group attach to get the full benefit).

### ***Batch considerations***

The procedures for batch and utilities also use STEPLIB, but the STEPLIB that they use does not contain the member name. An example of the statement to define the alias for batch and utilities is shown in Example 19-5.

#### *Example 19-5 ALIAS statement for batch STEPLIB*

---

```
DEFINE ALIAS -  
    (NAME(DB2HLQ1.DBP0.SDSNLOAD)      -  
     RELATE(DB2HLQ1.DBP0.RSU1404.SDSNLOAD)) -  
    CATALOG (ICFCAT.DB2HLQ)
```

DB2 STEPLIB has DB2HLQ1.DBP0.SDSNLOAD

---

Remember that batch and utilities are not sensitive to service levels within a DB2 version, so they can continue to use the current service level, RSU1404, until all the DB2 subsystems have moved to RSU1407. You then delete and define its ALIAS as shown in Example 19-6.

#### *Example 19-6 Switching the batch ALIAS to point to new service level*

---

```
DELETE -  
    DB2HLQ1.DBP0.SDSNLOAD -  
    ALIAS -  
    CATALOG(ICFCAT.DB2HLQ)
```

```
DEFINE ALIAS -  
    (NAME(DB2HLQ1.DBP0.SDSNLOAD)      -  
     RELATE(DB2HLQ1.DBP0.RSU1407.SDSNLOAD)) -  
    CATALOG(ICFCAT.DB2HLQ)
```

---

There is no need to quiesce batch (TSO, CAF) or utilities because they are not sensitive to maintenance level within a version.

This maintenance approach fulfills the fourth goal listed at the beginning of this section. That is, you can have one set of batch or utility JCL in PROCLIB and it can run on any system on which it is submitted.

### **Stored Procedure considerations**

Some basic premises are necessary to support the mixed release / maintenance coexistence strategy:

- ▶ An individual application environment has the scope of only one data sharing group.
- ▶ Each data sharing group needs its own application environments, with its own cataloged procedures in the procedure library and not in the application environment definition. Multiple environments are needed because certain options, like NUMTCB, can vary.

In the following example, the REXX stored procedure JCL is made available to all members of a data sharing group that can be running different releases or maintenance levels of DB2. The DB2SSN parameter is used as a part of the data set name, allowing redirection of the data set name based on the member SSN starting the Stored Procedure. When the procedure library is shared throughout the SYSPLEX, it allows for a single JCL definition for the PROC listed in the WLM definition as shown in Example 19-7.

*Example 19-7 Stored Procedure WLM application environment and cataloged procedure snippet*

---

```

Application Environment Name . : DBPOWLM_REXX
Description . . . . . WLM FOR REXX
Subsystem Type . . . . . DB2
Procedure Name . . . . . DBPOWLMR
Start Parameters . . . . . DB2SSN=&IWMSSNM

```

---

Limit on starting server address spaces for a subsystem instance

- ```

1  1. No limit
    2. Single address space per system
    3. Single address space per sysplex

```
- 

In the task cataloged procedure, there are the high-level qualifier, the group attach, and member name as shown here with the DB2SSN parameter:

```

//DBPOWLMR PROC RGN=OM,DB2SSN=,NUMTCB=1,APPLENV=DBPOWLM_REXX
//IEFPROC EXEC PGM=DSNX9WLM,REGION=&RGN,TIME=NOLIMIT,
//          PARM='&DB2SSN,&NUMTCB,&APPLENV'
//STEPLIB DD DISP=SHR,DSN=DB2HLQ1.DBP0.SDSNEXIT
//          DD DISP=SHR,DSN=DB2HLQ1.DBP0.&DB2SSN..SDSNLOAD
//          DD DISP=SHR,DSN=DB2HLQ1.DBP0.&DB2SSN..SDSNLOD2
//          DD DISP=SHR,DSN=DB2HLQ1.DBP0.RUNLIB.LOAD
//SYSEXEC DD DISP=SHR,DSN=xxxxxxxx.USER.REXX

```

The names of the stored procedures begin with the group attachment name to make it easy to identify the procedures for a specific group, but this is only a convention. If the stored procedure runs on DBP1, the library names resolve to DB2HLQ1.DBP0.DBP1.SDSNLOAD and so on, which is similar to that of the started tasks for the DB2 address spaces for member DBP1.

As the maintenance is rolled using the aliases, when the stored procedure is run on each member, it resolves to the maintenance level for that member.



## File system activities

When you roll the maintenance for your member DBP1, you must also switch your zFS to the mount point that matches the service level.

There are numerous ways to perform this kind of upgrade, but it is all based on providing the proper symlinks. The various PATH statements that are exported by the user should be independent of the z/OS system on which they run and the DB2 service level.

This example shows a method that was tested at the Washington Systems Center, where there are two data sharing groups: WSCDBP0 and WSCQUAL. The full example is shown in the “Example of DB2 maintenance and the zFS” on page 284. This section contains a simplified version with only one data sharing group and only two systems: SYSA and SYSB.

The data sharing group, WSCDBP0 can have members that have different service levels on the two systems. You want the users only to know they use the “db2dbp0” data sharing group from either SYSA or SYSB. They always export the same PATH, LIBPATH, or CLASSPATH string regardless of which system they are on. For example, regardless of whether they run on SYSA or SYSB you want them to use this command:

```
export PATH=/shared/db2dbp0:$PATH
```

Set the libraries up correctly so that when they run the program, it just works. This example uses a script called **red\_dbp0.sh**. When it is run, these tasks are completed:

- ▶ It displays the current service level (3rd qualifier of the D2 library names).
- ▶ It sets PATH, LIBPATH, and CLASSPATH.
- ▶ It sets STEPLIB concatenation so the subsequent **echo \$STEPLIB** command displays it, validating the method.

The file system is shared sysplex wide. DB2 maintenance is not tied to z/OS maintenance, but is treated separately:

- ▶ A mount point is created off the root directory, called /shared.
- ▶ In that file system, among others, there are two mount points: **db2\_I1** and **db2\_I2**:
  - DB2HQ1.DBP0.RSU1404.DB2.ZFS is mounted at “db2\_I1”. It contains the current level.
  - DB2HQ1.DBP0.RSU1407.DB2.ZFS is mounted at “db2\_I2”. It contains the new level.
- ▶ You want the instance of “db2dbp0” on SYSA to use the current version of the code base and the instance of “db2dbp0” on SYSB to also use the current version of the code.
- ▶ Create a number of symlinks that resolve to the correct level of code:
  - In the /SYSA file system, create two symlinks:

```
jrb_11 --> /shared/jrb_11
jrb_12 --> /shared/jrb_12
```
  - Do the same in the /SYSB file system.
  - In the /SYSA file system, create the symlink “which\_dbp0” and point it at “jrb\_11”.
  - In the /SYSB file system, create the symlink “which\_dbp0” and point it at “jrb\_11”.

Now, make /shared/db2dbp0 resolve correctly depending on the system that is being used. In the /shared file system, create a symlink, “db2dbp0” and point it at /\$SYSNAME/which\_dbp0. \$SYSNAME is a z/OS system symbol that denotes the system name and is set at IPL time. It resolves to the file systems, /SYSA or /SYSB.

All the users know is that their information is in /shared/db2dbp0, no matter which of the two systems are used.

When DBP1 is upgraded in service, it is only necessary to remove and re-create the symlink “which\_dbp0” and point it at “jrb\_l2” on /SYSB.

If the client is running on SYSB, it continues to use “jrb\_l1”.

When the ALIAS for DBP1 is deleted and defined, the symlink that pointed to “jrb\_l1” is deleted and defined for “jrb\_l2”.

Now any job running on DBP1 for WSCDBP0 will have the zFS at the same level as the DB2 code. If errors arise in the maintenance, it is a matter of reverting the symlink for “jrb\_l2” to “jrb\_l1” concurrent with redefinition of the corresponding ALIAS for the DB2 libraries.

The explanation given here is brief. Appendix B, “DB2 UNIX file system maintenance considerations” on page 283 illustrates it more completely, and includes the symlinks necessary when there is more than one data sharing group on each system.

### ***Java stored procedures considerations***

Everything that is stated in “Stored Procedure considerations” on page 222 applies to Java stored procedures. The WLM address space start procedure for Java routines requires extra DD statements that other routines do not need. There are a few more considerations, centering on the JAVAENV data set. Example 19-8 is supplied by DB2. It is called xxxxWLMJ, and is found in job DSNTIJMV in SDSNDAMP.

*Example 19-8 JAVAENV for xxxxWLMJ as distributed with DB2*

---

```
ENVAR("_CEE_ENVFILE=/usr/lpp/db2b10/base/classes/dsnenvfile.txt",
      "DB2_BASE=/usr/lpp/db2b10/base",
      "JCC_HOME=/usr/lpp/db2b10/jdbc",
      "JAVA_HOME=/usr/lpp/java150/J5.0",
      "JVMPROPS=/usr/lpp/java/properties/dsnjvmssp"),
MSGFILE(JSPDEBUG,, ,ENQ),
XPLINK(ON)
```

---

Example 19-9 is an example of what was done in the example environment and might not be universally applicable. It shows the JAVAENV data set in DBP0WLMJ that is running in the WSCDBP0 group:

*Example 19-9 JAVAENV data set in DBP0WLMJ*

---

```
ENVAR("_CEE_ENVFILE=/shared/db2_config/wscdbp0/etc/dbp0envfile.txt",
      "DB2_BASE=/shared/db2dbp0/base",
      "JCC_HOME=/shared/db2dbp0/jdbc",
      "JAVA_HOME=/shared/java/J5.0",
      "JVMPROPS=/shared/db2_config/wscdbp0/etc/dbp0jvmssp"),MSGFILE(JSPDEBUG),XPLINK(ON
),
RPT0(ON)
```

---

**Note:** The \_\_JVMPROPS is not used. JVMPROPS actually is in dbp0envfile.txt. The two underscore characters are placed in front to keep it from being used. The line did not have to be removed and it is kept only for reference. If you choose not to use this form, the last lines look like this:

```
MSGFILE(JSPDEBUG),XPLINK(ON),  
RPT0(ON)
```

Both ENVAR and JVMPROPS share a common directory path. Only the contents of the files in the etc directory are different.

```
wsc1:/u/judyrb  
-> cd /shared/db2_config/wscdbp0/etc  
wsc1:/shared/db2_config/wscdbp0/etc  
-> ls -al  
total 48  
drwxr-xr-x  2 BROWN SYS1      8192 Jun 27  2011 .  
drwxr-xr-x  4 HUTCH SYS1      8192 Jun 27  2011 ..  
-rw-r--r--  1 BROWN SYS1       303 Jun 27  2011 dbp0envfile.txt  
-rw-r--r--  1 BROWN SYS1       125 Jun 28  2011 dbp0jvmmsp  
wsc1:/shared/db2_config/wscdbp0/etc
```

The DB2\_BASE and JCC\_HOME keywords have “db2dbp0” substituted for the base and JDBC paths, so the users do not have to know the actual maintenance level.

Example 19-10 shows the contents of the two files.

*Example 19-10 dbp0envfile.txt in ENVAR in JAVAENV for DBP0WLMJ*

---

```
-> cat dbp0envfile.txt  
CLASSPATH=/usr/include/java_classes/gx1japi.jar:/usr/include/java_classes/gx1jo  
srgImpl.jar  
LIBPATH=/usr/lib/java_runtime  
STEPLIB=DB2HLQ1.DBP0.SDSNLOAD  
JVMPROPS=/shared/db2_config/wscdbp0/etc/dbp0jvmmsp  
wsc1:/shared/db2_config/wscdbp0/etc
```

---

The only DB2 related library is STEPLIB. It can be treated like batch as covered in “Batch considerations” on page 221. The IBM Data Server Driver for JDBC and SQLJ is not sensitive to service level in SDSNLOAD. The driver is sensitive to service level in SDSNLOAD2 and must match it exactly. As stated, the changing of the ALIAS and the symlinks in the file system must occur together.

For the JVMPROPS it looks as shown in Example 19-11.

*Example 19-11 JVMPROPS in JAVAENV data set for DBP0WLMJ*

---

```
-> cat dbp0jvmmsp  
# sets the initial size of middleware heap within non-system heap  
-Xms64M  
# Sets the maximum size of nonsystem heap  
-Xmx128M  
wsc1:/shared/db2_config/wscdbp0/etc
```

---

JVMPROPS contains only system related JVM properties, not DB2 libraries. There are no maintenance concerns.

## 19.1.2 Version to version upgrade/fallback

The upgrade of your DB2 data sharing group can be accomplished without shutting down the entire group. The upgrade process for each DB2 version is specific to that version and is described in detail in the *DB2 for z/OS Installation and Migration Guide*. Normally upgrade occurs from one version to the next. Rarely, “skip release” upgrade can occur, as happened in DB2 10, where upgrade was supported from DB2 Version 8 (and DB2 9) to DB2 10. Upgrade can only occur from DB2 10 to DB2 11.

You can attach to multiple versions of DB2 with your existing TSO or CAF logon procedures without changing the load libraries for your applications. After you upgrade completely to the latest level of DB2, you must update those procedures and jobs to point to the latest level of DB2 load libraries.

There is no need to change the cataloged procedures if the library names chosen are not version-specific. That is, use DB2HLQ1 versus DSN1010 as the qualifier for SDSNLOAD and the other data sets. Typically you keep the group active and stop one member with the old version and restart it with the new version. When you have done so, the state of that member is conversion mode (CM) and the entire group is in “coexistence mode”. It is best to remain in coexistence mode no more than two weeks. Although there is full IBM support for all modes, the longer you remain in coexistence mode, the more likely you will encounter problems specific to coexistence. Service is provided but will likely delay your upgrade to complete CM mode, in which the problem would not have occurred at all. During that time, you stop and start each member using the new libraries. When all members are in the new version, your group is no longer in coexistence mode. Stay in CM mode for several months for completion of month end, quarter end, and other special cycles. You can return to the previous version at any time in CM. The DB2 code is at the new version, and some new capabilities can be implemented, but no new SQL can occur. Some features of the new version can be implemented when static packages are rebound because access paths for new features are eligible. Rebind high-use packages in CM mode to gain more benefits from the new version.

You will go forward to the next stage of migration: Enabling-new-function mode (ENFM). After you enter ENFM, you cannot return to the previous version of DB2. This phase might be short or long, and is focused on the changes that are necessary to the DB2 catalog/directory for that version. Changes are usually implemented by using REORG SHRLEVEL CHANGE of the affected catalog table space. It is not necessary to complete the ENFM job, called DSNTIJEN, in one pass. It saves the last step completed and when it is restarted, it will pick up at the next step. There are restrictions when the DSNTIJEN job is running. Business transactions, both static and dynamic *and that frequently commit* can run. Avoid BINDs, DDL, and utilities while catalog table spaces are being processed. Remaining in ENFM is acceptable, but no new SQL can be used. You can return to CM mode, in which case the state when you do a DISPLAY GROUP shows CM\*, meaning that your data sharing group has achieved a higher level in the past.

When the DSNTIJEN upgrade job has completed successfully, you are ready for new function mode (NFM). The job that you run from SDSNSAMP is DSNTIJNF. You have now reached the goal of the upgrade, and you can use all of the capabilities of the new version, including SQL. However, you can return to ENFM if you want. Similar to CM\*, your group state shows as ENFM\*, and any NFM function that has been used is now “frozen”. It cannot be used again until you return to NFM. The *DB2 for z/OS Installation and Migration Guide* covers the upgrade and fallback process in detail for DB2 versions.

### 19.1.3 DSNZPARM changes

The DSNZPARM control block is specific to each DB2 member. Most of the DSNZPARMs found in SDSNSAMP(DSNTIJUZ) can be implemented without recycling DB2. All new DSNZPARMs beginning in DB2 Version 8 are online changeable.

**Note:** Of several hundred DSNZPARMs, only about 50 are not online changeable.

Follow the instructions under the **-SET SYSPARM DB2** command to update, produce a new subsystem parameter DSNZPARM module, and load it into DB2 while it is active.

See the *DB2 for z/OS Installation and Migration Guide* for a list of those keyword values. They are found under a topic similar to “Directory of subsystem parameters and application default values” under a column called “Update online”.

### 19.1.4 Convert BSDS to Extended RBA/LRSN in DB2 11

The bootstrap data set (BSDS) tracks its own log records and can read log records of other data sharing members. Up to DB2 11, the DB2 log had a continuously ascending relative byte address (RBA) of 6 bytes. The data sharing group-wide time stamp is the log record sequence number (LRSN) of 6 bytes. DB2 11 began using *10-byte* RBA and LRSN values because clients had either used up the former 6-byte limit that had been in effect for 30 years or were approaching it. With faster processors, consolidation of DB2 members that occurred in DB2 10, and heavy, sustained logging rates, DB2 can exhaust the 6-byte RBA.

The extended capability expands the limit of the DB2 log for each member to 1 yottabyte (2<sup>80</sup> or 1,208,925,819,614,629,174,706,176 bytes), but both logs and data need to be “primed” to use it. This conversion of the BSDS and the data pages can occur non-disruptively in DB2 11, and can occur in any order. For more information about data conversion to the Extended page format, see “Change DB2 objects from Basic format to Extended format” on page 242.

If the limit was reached by a client before DB2 11, manual recovery actions were necessary to reset the RBA to zeros. In data sharing, the affected member was shut down and a new member was started. In non-data sharing, the outage was lengthy because each table space had to be reset individually to zeros by using a special form of the COPY utility. Both conditions are considered to be emergency type outages that are unplanned.

Warning messages are issued on every active log switch, beginning about a year before the end of the log is reached.

The 6-byte LRSN/RBA continues to be used in DB2 11 CM, and conversion of the BSDS cannot occur until the client is in NFM. Any time after that, the client can shut down a member of the data sharing group, and convert the BSDS using the stand-alone utility DSNJCNVT, which is implemented using an installation job found in SDSNSAMP(DSNTIJCB). The outage should take about a minute. The conversion can be rolled through one member at any time because BSDS can be read in either 6- or 10- byte mode.

**Note:** If the client is not close to the end of the log or the LRSN, this action is entirely optional. However, DB2 11 code assumes that all RBA/LRSN values are 10-bytes. Any that are still 6-byte values are dynamically converted to 10 bytes temporarily, so conversion is a good idea

### 19.1.5 Dynamic addition of active logs

If you wanted to add active logs to an existing DB2 subsystem before DB2 10, it was necessary to stop DB2 to run DSNU0003, Change Log Inventory, which then added the log.

Beginning in DB2 10, you just define the data set to IDCAMS and then issue the **DB2 -SET LOG** command, specifying the name of the data set you created and whether it is COPY1 or COPY2. However, for best performance format the new log after defining it using the DSNJLOGF utility. Because dual logging is normally used, you should go through this process twice, so that both copies are added. As of DB2 11, the maximum number of active logs is 93 per DB2 member or subsystem.

To add active logs, complete these steps:

1. Define the new active log COPY 1 with IDCAMS
2. Define the new active log COPY2 with IDCAMS
3. Preformat new active log COPY1 using DSNJLOGF
4. Preformat new active log COPY2 using DSNJLOGF
5. Issue -SET LOG command for COPY1
6. Issue -SET LOG command for COPY2

## 19.2 Database changes

This section is of more interest to the DBA community. Availability of the subsystem is good, but the data must also be available. Planned outages exist for both maintenance of the DB2 subsystem and also for the table spaces, partitions of table spaces, and indexes over the data. Changes to the underlying data can involve long outages if the data must be dropped and re-created, or might be much shorter if only a REORG is necessary to implement the changes. Figure 19-1 shows the actions that are required before DB2 Version 8 to change a column definition.

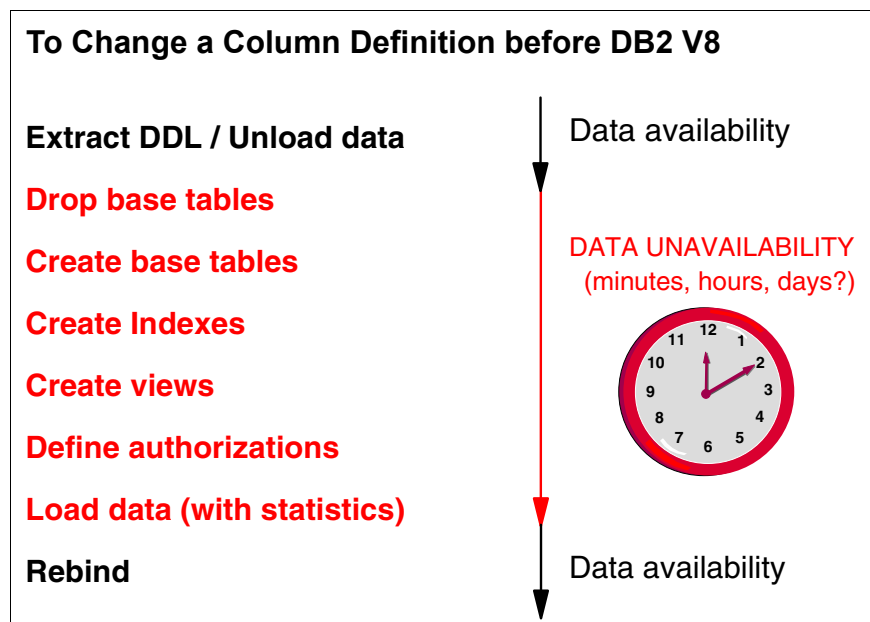


Figure 19-1 Changing a column definition before DB2 Version 8

For many changes to the underlying objects that use Data Definition Language (DDL), it was necessary to unload the data, drop the objects, define the objects, grant authorizations, reload the data (with statistics), and bind the packages. The outage could take several hours, particularly if the data included many partitions (up to 4096) in a table space. In DB2 for z/OS Version 8, “schema evolution” was introduced as a set of capabilities intended to be implemented over multiple DB2 versions. The idea was simple: Allow schema changes to be implemented by only a REORG at the client’s convenience. All existing objects, package, and authorizations would remain. A REORG with SHRLEVEL CHANGE allows updates to occur on the unchanged objects while the data are being changed behind the scenes on the shadow objects. A brief outage occurs during the SWITCH phase of REORG, at the end of which the changed objects become active. The multiple hour outage is reduced to a few minutes or less.

The schema evolution changes are normally started by an ALTER, which is a DDL statement. In most cases the objects (table space, partition, index) are available for access until the REORG occurs. With DB2 9, depending on which change you made, the change can be implemented immediately by updating the attributes in the catalog and directory, and the page sets might be placed in pending states such as the restrictive REORG-pending (REORP), the advisory REORG-pending (AREO\*), or Rebuild-pending (RBDP) with the associated consequences, such as indexes not being considered by the optimizer.

With DB2 10, in addition to new online schema changes, some of the changes that might restrict access to the changed database objects cause no immediate action and do not materialize immediately. DB2 10 introduces the concept of pending changes (which are materialized only when you reorganize the altered objects) and the new associated advisory pending state (AREOR), which is described in the next section. With DB2 10, objects in the REORP restrictive state or AREOR advisory pending state are reset by REORG with SHRLEVEL REFERENCE or CHANGE, instead of the restriction to just NONE in DB2 9.

## 19.2.1 DB2 for z/OS Version 8

This version is out of service as of April 30, 2012, but the schema evolution items are listed. For more information, see *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079.

- ▶ Add/drop clustering index
- ▶ Index padding change
- ▶ Add at partition
- ▶ Rotate partitions
- ▶ Rebalance partitions
- ▶ Change data types

## 19.2.2 DB2 for z/OS 9

This version is out of service as of June 27, 2014. For more information, see *DB2 9 for z/OS Technical Overview*, SG24-7330. Table 19-2 shows the availability enhancements.

Table 19-2 Availability enhancements in DB2 9

| Change        | Described in                  | Notes |
|---------------|-------------------------------|-------|
| Rename column | “Rename a column” on page 230 |       |
| Rename index  | “Rename index” on page 230    |       |

| Change                        | Described in                                         | Notes                                                 |
|-------------------------------|------------------------------------------------------|-------------------------------------------------------|
| Modify EARLY code without IPL | "Modify EARLY code without IPL" on page 231          |                                                       |
| CLONE tables                  | "CLONE tables" on page 231                           |                                                       |
| No BUILD2 phase for NPIs      | "Eliminate BUILD2 phase of online REORG" on page 232 | However, see the DB2 11 reinstatement cross-reference |

## Rename a column

Continuing with Online Schema changes from DB2 Version 8, under some circumstances in DB2 9 you can rename columns of an existing table without having to drop and re-create the object. A new clause is added to the ALTER TABLE statement. If ALTER TABLE RENAME COLUMN succeeds, all the existing objects associated with that column, such as tables, indexes, RI, auxiliary tables, auxiliary indexes, foreign keys, and statistic information function the same way as before renaming the column.

RENAME COLUMN results in the invalidation of any plan or package that depends on the table whose column is being renamed. Any attempt to run the invalidated plan or package triggers an automatic rebinding of the plan or package.

The simple syntax for altering a column name is:

```
ALTER TABLE tablename RENAME COLUMN oldcolumnname TO newcolumnname
```

Keep in mind these restrictions when you plan to rename a column:

- ▶ The RENAME COLUMN keyword cannot be used in one statement with any other ALTER TABLE options.
- ▶ The RENAME COLUMN keyword cannot be repeated within one ALTER TABLE statement.

ALTER TABLE RENAME COLUMN is not allowed in these situations:

- ▶ When the source column is referenced in a view. If you want to rename a column for a table that has views defined on it, you must drop the views first. You can freely rename all unreferenced columns
- ▶ For a table with a trigger. You must drop the trigger, rename the column, and re-create the trigger afterward. This is the case for all columns (even those columns that are not specified in the trigger cannot be renamed).
- ▶ If the table is referenced in a materialized query table (MQT).
- ▶ If the source column has a check constraint or a field procedure defined on it.
- ▶ If the table containing the column has a valid or edit procedure defined on it. You cannot drop edit and field procedures (you can never use the ALTER TABLE statement to rename those columns).
- ▶ If the source column has an index defined on it. You must drop all indexes that are referencing the column to be able to rename the column.
- ▶ If there is an index on expression or spatial index (depends on the column).

## Rename index

The capability to rename an index without dropping and re-creating was introduced in DB2 9. You specify to RENAME old-index-name TO new-index-name. The qualifier of the source index name is used to qualify the new name for the index.



As for any other object, an index is unambiguously identified within a database by its OBID. Renaming the index does not change this number. Plans and packages identify indexes used in their access paths by those IDs. As a result, your plans and packages are not invalidated when you rename an index.

However, the index names are used for the identification of indexes that are used by statements that are stored in the dynamic statement cache. Therefore, statements in the dynamic cache are invalidated if they ought to use the index that you renamed.

## Modify EARLY code without IPL

EARLY code is normally upward and downward compatible by one version. Ever since DB2 was first developed over 30 years ago, the activation of EARLY code in a new version of DB2 was cumbersome because an IPL was necessary for this activation. It is no longer necessary. You can use the command **-REFRESH DB2,EARLY** to reload the code and rebuild the EARLY control block. The command must be issued when the DB2 member s for that system are shut down. Otherwise the command fails. Because this command only has member scope, you must issue the **REFRESH** command for every instance where a unique recognition character for the **-START** command has been defined. For more information about early code, see “SDSNLINK” on page 220.

**Note:** The **-REFRESH** command works if you copy the individual modules into an existing LINKLIST library. If you replace or add a library to the LINKLIST, an IPL is required.

## CLONE tables

Clone tables are available only for universal table spaces (UTS), either partitioned by range (PBR) or partitioned by group (PBG).

Users expressed the need for a function that is similar to an online LOAD REPLACE so they can reload table data multiple times a day while still being able to access the existing data at least with read-only operations. Clone table support allows you to generate a table with the exact same attributes as a table that exists on the current server. This new table is called a clone table of the base table, and is created in the same table space as the base table. After the clone table is created, you can independently work with it (you can load it or insert rows into it).

The clone table is not only structurally identical to the base table (that is, it has the same number of columns, column names, data types, check constraints), but it is also created with the same indexes, before triggers, LOB objects, and so on. DB2 creates the clone objects automatically when it creates the clone table. Although the clone table itself has a different name and might have a different schema associated to it, the clone table objects are referred to by the same names as those that are used for the base table objects (that is, the base and clone table share all object descriptions). You can create a clone table on an existing base table using the **ALTER TABLE ADD CLONE** syntax.

The table space also must be DB-managed rather than user-managed.

**Attention:** User-managed objects, defined by you with IDCAMS are phasing out quickly, particularly with the SMS-management of the DB2 catalog/directory in DB2 10.

When you create clone tables, the whole purpose is to switch from the original base table to your newly created and maybe-reloaded clone table. Use the SQL statement **EXCHANGE DATA BETWEEN TABLE table-name1 AND table-name2** to initiate the swap. For this statement, it does not matter which table is the clone table and which table is the base table.

No data movement takes place when the tables are exchanged. Only the instance numbers that point to the base or clone table change. What originally was the pre-exchange clone table data instance number becomes the base table data instance number and vice-versa after the exchange completes.

This capability provides an Online Load Replace function. The exchange is made at the table level. It is not possible to exchange at the partition level.

### Eliminate BUILD2 phase of online REORG

When reorganizing a single partition of a table space with one or more non-partitioning indexes (NPIs), the phases of REORG SHRLEVEL CHANGE are shown in Figure 19-2. Data are still available during the reload and rebuild phases. However, reorganization by partition suffers from a relatively long outage when NPIs are defined on the table.

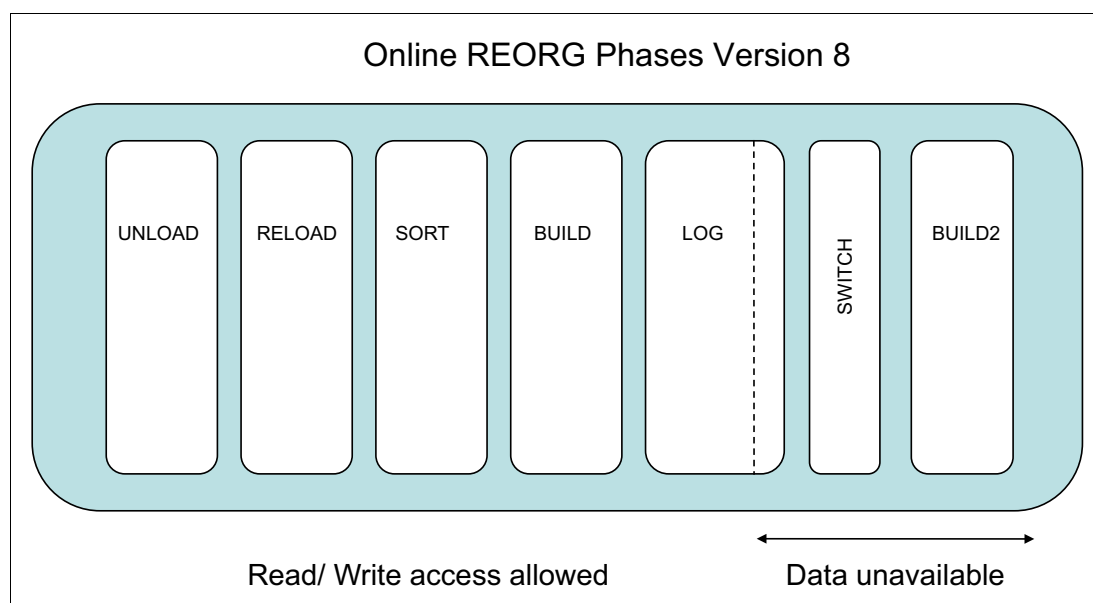


Figure 19-2 Online REORG phases in V8 for a partition and NPIs

During OLR before V9, the NPIs had to be updated with the new RIDS (row pointers) for the data of the partition being reorganized during the BUILD2 phase. The availability outage to the index can be significant. The NPI became group buffer pool (GBP) dependent as well and the RID updates might flood the GBP.

DB2 V9 removes the need for a BUILD2 phase, by reorganizing the whole of the NPI. The reorganized NPIs are then switched in the SWITCH phase along with the other data sets. The only outage is during the relatively short SWITCH phase, as shown in Figure 19-3. There is no GBP-dependency either because the changes occur on the shadow data sets.

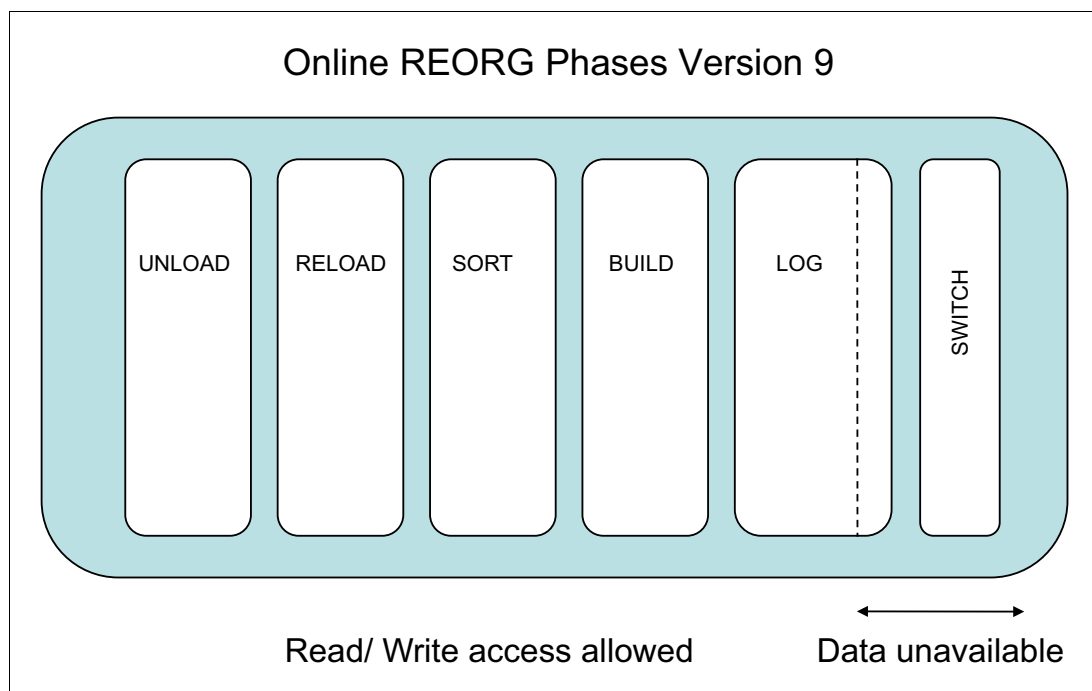


Figure 19-3 Online REORG phases in V9 for a partition and NPIs

The parameter SHRLEVEL controls the execution of REORG with these options:

- ▶ SHRLEVEL NONE: No access
- ▶ SHRLEVEL REFERENCE: Allows only readers
- ▶ SHRLEVEL CHANGE: Allows readers and writers

REORG SHRLEVEL REFERENCE also eliminated the BUILD2 phase.

Generally, reorg multiple partitions together rather than serially, as each single partition reorg also reorgs the entire NPI.

Specify Reorg parts 200-221 in DB2 9:

- ▶ Specify Reorg part 200-221, so the NPIs are reorged only once. This is a better choice than three REORG jobs with 200-204, 213, 218-221 because each of the three reorg jobs would also reorg the NPIs.
- ▶ All ranges must be continuous

### 19.2.3 DB2 for z/OS 10

DB2 for z/OS 10 implemented online schema changes more autonomically.

The ALTERs for the schema changes for an object are now stored in a new catalog table, called SYSIBM.SYSPENDINGDDL. The new advisory REORG pending state (AREOR) is set for the object. Some ALTERs can be implemented in the same REORG. At a time of your choosing, REORG the table space or index or do a LOAD REPLACE to materialize the changes for the entire table space or index space. If you do not want to implement the

changes, ALTER the object, specifying DROP PENDING CHANGES and all pending changes are eliminated. If DB2 requires your schema change to be a pending definition change, semantic validation and authorization checking are done immediately when you run the ALTER statement. Table 19-3 shows the availability enhancements for DB2 10.

**Note:** You must REORG with SHRLEVEL options either REFERENCE or CHANGE. NONE does not materialize pending changes.

For more information about each of the listed enhancements, see *DB2 10 for z/OS Technical Overview*, SG24-7892.

Table 19-3 Availability enhancements for DB2 10

| Change                           | Described in                                                                    | Notes (optional)               |
|----------------------------------|---------------------------------------------------------------------------------|--------------------------------|
| Page size of data                | "Alter page size of table space" on page 235                                    |                                |
| Page size of index               | "Alter page size for index" on page 236                                         |                                |
| DSSIZE                           | "Alter DSSIZE" on page 236                                                      |                                |
| SEGSIZE of UTS                   | "Alter SEGSIZE of UTS" on page 237                                              |                                |
| Table space type to UTS          | "Alter the table space type to UTS" on page 237                                 |                                |
| MEMBER CLUSTER                   | "Alter MEMBER CLUSTER attribute for UTS" on page 238                            |                                |
| UTS to Hash table space          | "Convert from a UTS to hash table space" on page 238                            |                                |
| Hash table space to UTS          | "Convert from hash organization to UTS" on page 238                             |                                |
| REORG availability               | "REORG improvements for availability" on page 239                               |                                |
| Consistent copies without outage | "Create consistent image copies of multiple objects without outage" on page 240 |                                |
| CHECK DSNZPARM default           | "CHECK increased availability" on page 241                                      |                                |
| DDF additions to CDB             | "Add or update CDB without stopping DDF" on page 241                            |                                |
| DDF Location Alias               | "Change DDF location aliases dynamically" on page 241                           |                                |
| Add active log dynamically       | 19.1.5, "Dynamic addition of active logs" on page 228                           | Run twice for dual active logs |

Before DB2 V8, a change to a database definition almost always required that you unload the data, drop the object, re-create the object, reestablish authorization on the object, re-create the views, and then reload the data. DB2 V8 introduced ALTER statements to change database object attributes online. DB2 9 added more possible changes. DB2 10 includes

many more ALTERs that allow you to convert the structure of the table space and change attributes.

**Key point:** All the DB2 10 ALTER options apply to universal table spaces (UTS).

The Universal Table Space (UTS) was introduced in DB2 9 with two separate organizations: Partition-by-growth (PBG) and partition-by-range (PBR). Existing table spaces cannot be converted to UTS without an unload/drop/create and load of the UTS. The conversion can now occur with an ALTER and REORG. After the objects are UTS, multiple changes can occur at the same time. To be able to alter these other attributes, create all new table spaces as UTS, and as time and resources permit, convert all existing table spaces to UTS.

You might want to convert the structure from simple table spaces, which are deprecated since DB2 9. Or you might want to convert the segmented and partitioned table spaces to the UTS structure to use its characteristics. Figure 19-4 shows the possible conversions. Be sure to observe the “one way” signs.

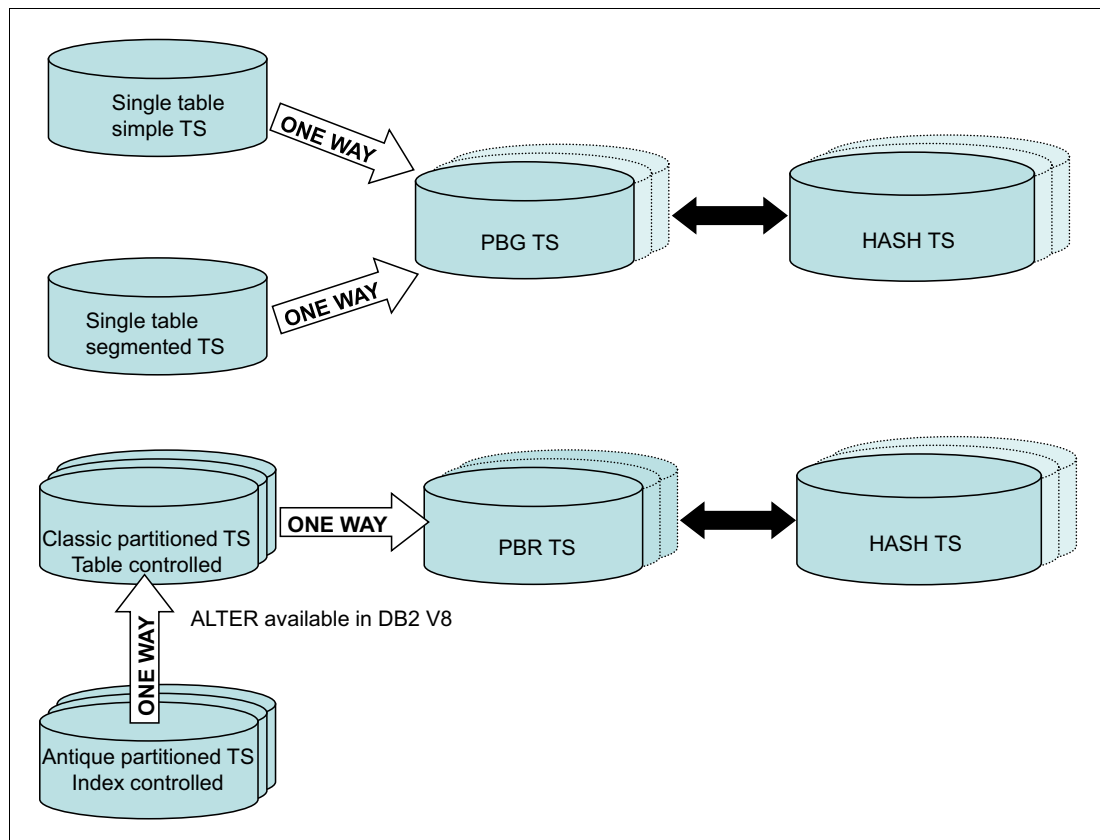


Figure 19-4 Possible table space type conversions

### Alter page size of table space

This capability is for UTS or LOB only.

Before DB2 10, you could change the buffer pool of a table space, but you could do so only if both the original and the new page size were the same. With DB2 10, you can change the page size to either a smaller or larger new page size using the BUFFERPOOL keyword on an ALTER TABLESPACE statement. The BUFFERPOOL must already be defined.

This ALTER statement is not materialized immediately because this is a change to a buffer pool with a different page size. Instead, a row is added to SYSIBM.SYSPENDINGDDL that shows exactly which change is pending, what will be materialized to the catalog, and your page set if you ran a REORG table space now.

The table space for which you increase the page size, depending on the average size of the row, can lead to a reduction or increase of total space needed after REORG is run. If the space increases, you might consider changes to the DSSIZE as well (see “Alter DSSIZE” on page 236).

If you have LOB columns in the table for which you increase the page size, consider the possible impact on LOB table spaces because auxiliary objects might be implicitly created for the newly grown partitions. It does not matter whether your table space was created implicitly or explicitly. The new LOB objects inherit the buffer pool attribute from the existing LOB objects.

If the space decreases, keep in mind that for partition-by-growth table spaces empty partitions are not removed. After a partition is allocated and registered in the DB2 catalog, it remains defined even after REORG or LOAD. An empty partition is allocated with its primary allocation size.

If you try to ALTER the page size to a value smaller than the row length, you receive SQLCODE -670 and the change request is not accepted.

### **Alter page size for index**

Before DB2 10, you can change page sizes, assigning a different buffer pool with a different page size to an index. However, you cannot implement this change immediately. Therefore, the index is placed in rebuild pending (RBPDP) state.

With DB2 10, you can change the page size of an index, and the index is no longer placed in an RBPDP state if the index is of one of the following types:

- ▶ The index is defined on a table that is contained in UTS.
- ▶ The index is defined on an XML table that is associated with a base UTS.
- ▶ The index is defined on an AUXILIARY table that is associated with a base universal table space.

If the index is of one of these types, the ALTER is considered as a pending change. The index is placed into AREOR state, which improves availability.

### **Alter DSSIZE**

These factors affect the choice of DSSIZE:

- ▶ The specification or not of the LARGE option
- ▶ The specification of DSSIZE originally
- ▶ The number of partitions
- ▶ The value of DSSIZE in SYSTABLESPACE
- ▶ Maximum size per partition

For a more detailed description of these issues, see the ALTER TABLE SQL statement in *DB2 9 for z/OS SQL Reference*, SC19-2983.

Table 19-4 summarizes the DSSIZE value considerations.

Table 19-4 Maximum DSSIZE depending on table space attributes

| LARGE option used? | DSSIZE specified | Number of partitions | Value of DSSIZE in SYSTABLESPACE                                                                          | Maximum size per partition |
|--------------------|------------------|----------------------|-----------------------------------------------------------------------------------------------------------|----------------------------|
| NO                 | NO               | 1 - 16               | 0                                                                                                         | 4 GB                       |
| NO                 | No               | 17 - 32              | 0                                                                                                         | 2 GB                       |
| NO                 | NO               | 33 - 64              | 0                                                                                                         | 1 GB                       |
| NO                 | NO               | 65 - 4096            | 0                                                                                                         | 4 GB                       |
| NO                 | NO               | 255 - 4096           | > 0, the DSSIZE equals the page size of the table space                                                   | Same as DSSIZE             |
| YES                | NO               | 1 - 4096             | 0                                                                                                         | 4 GB                       |
| NO                 | YES              | 1 - 4096             | > 0, the DSSIZE must be valid, depending on the number of partitions and the page size of the table space | Same as DSSIZE             |

## Alter SEGSIZE of UTS

Before DB2 10, altering the segment size of a table space was not allowed. With DB2 10, you can alter the segment size of UTS. When you want to use this option, **ALTER TABLESPACE...SEGSIZE** does not allow any additional options in this statement. However, you can issue several ALTER TABLESPACE statements in a sequence such as:

```
ALTER TABLESPACE ITSODB.TS1 BUFFERPOOL BP16K0;
ALTER TABLESPACE ITSODB.TS1 DSSIZE 4 G ;
```

Each ALTER then leads to a new row in SYSIBM.SYSPENDINGDDL.

## Alter the table space type to UTS

The following table spaces can be changed to UTS but cannot be changed back to the original table space type:

- ▶ *Single* simple table space to UTS PBG
- ▶ *Single* segmented table space to UTS PBG
- ▶ Classic partitioned table space to UTS PBR

### Conversion of SINGLE simple or segmented table space to UTS PBG

For the single table spaces, in DB2 10, you use the MAXPARTITIONS extension to the ALTER TABLESPACE statement. If you specify MAXPARTITIONS and the table space does not yet exist physically, the change takes effect immediately if no other changes are pending for this specific table space. If the table space data sets are already defined, this change is another pending definition change that is materialized by DB2 when you run the next REORG table space.

### Conversion of classic partitioned table space to UTS PBR

With DB2 10, the ALTER TABLESPACE statement allows you to change the table space definition from a classic partitioned table space to a UTS range-partitioned table space. The keyword to use is SEGSIZE. When you specify SEGSIZE, this keyword is mutually exclusive with the other keywords on your ALTER TABLESPACE statement. It is handled as an immediate change only if the table space data sets are not yet defined. If the table space is

allocated physically, that is, the VSAM clusters exist, then this change is handled as a pending definition change, and your table space is placed in AREOR state.

The following conditions apply:

- ▶ The MEMBER CLUSTER attribute is inherited from the original table space. DB2 9 does not support MEMBER CLUSTER for UTS. The fact that MEMBER CLUSTER is supported in DB2 10 for UTS makes this behavior possible.
- ▶ The number of partitions is inherited from the original table space.
- ▶ If the value for DSSIZE was not specified upon creation of the original table space and therefore the current setting is 0, this is changed to the maximum possible DSSIZE. If DSSIZE was specified already, the range-partitioned table space inherits the setting from the original table space.

### **Alter MEMBER CLUSTER attribute for UTS**

MEMBER CLUSTER is a table space option that allows reduced contention in a heavy INSERT application in data sharing environment. DB2 9 does not support MEMBER CLUSTER for a UTS. DB2 10 allows MEMBER CLUSTER for both PBG UTS and PBR UTS. You can ALTER an existing UTS to add the MEMBER CLUSTER option. For example, the following statement adds a row to SYSIBM.SYSPENDINGDDL and places the table space in AREOR advisory state:

```
ALTER TABLESPACE ITS0DB.TS2 MEMBER CLUSTER YES;
```

### **Convert from a UTS to hash table space**

If you want to convert your PBG or PBR table space to hash, a few restrictions apply to both conversions:

- ▶ The hash column or columns must be defined as NOT NULL.
- ▶ The table to be converted cannot be APPEND(YES) or MEMBER CLUSTER.
- ▶ The table to be converted must be in a UTS as either PBG or PBR.
- ▶ The table cannot be a global temporary table.

Assume that your analysis shows that the hash organization is beneficial for your application, and you want to change the organization of your PBG to hash. In this example, you have PBG table space with one unique index on column EMPNO.

The syntax to change the organization of the table space in the example, HASHDB.HASHTS, from PBG to hash is as follows:

```
ALTER TABLE HASHTB ADD ORGANIZE BY HASH  
UNIQUE (EMPNO) HASH SPACE 1 M;
```

After issuing this SQL statement, you receive SQL code +610, which informs you that your table space is placed in AREOR state. In addition, a hash overflow index is created, which is not built immediately but that is placed into Page Set Rebuild pending (PSRBD) restrictive state. The information is put into SYSIBM.SYSPENDINGDDL for resolution by REORG.

### **Convert from hash organization to UTS**

If you determine that you do not benefit from the HASH organization, you can convert an existing hash table space to PBG or PBR UTS. The ALTER TABLE syntax includes the DROP ORGANIZATION key word. When you issue the ALTER TABLE statement, the hash overflow index is immediately dropped and the table space is placed in REORG pending (REORP) state. This state is a restrictive state. You must run a REORG before you can use it. The REORG must either be SHRLEVEL REFERENCE or SHRLEVEL CHANGE.



## **REORG improvements for availability**

In this example, *REORG* means REORG with SHRLEVEL CHANGE or REFERENCE as they provide most availability. SHRLEVEL NONE, which provides no availability during the entire REORG, is being deprecated. It also cannot be used for materializing pending DDL. The following enhancements are available in new-function mode (NFM):

- ▶ SHRLEVEL CHANGE for all catalog/directory table spaces
- ▶ SHRLEVEL CHANGE for LOBs
- ▶ AUX column of base table for improved LOB handling
- ▶ Multiple partition ranges can be specified in one REORG
- ▶ Reduce need for REORG INDEX with index leaf page list prefetch
- ▶ Reduce need for REORG with compress on insert

### ***SHRLEVEL CHANGE for all catalog/directory table spaces***

With all the links removed from the DB2 catalog and directory by DB2 10, true online reorg that allows readers and writers can be performed for all the catalog and directory. The one exception is SYSUTILX, which cannot ever be reorged, as it tracks utilities as they are running itself.

### ***SHRLEVEL CHANGE for LOBs***

Up to DB2 Version 8, the only possible SHRLEVEL option for the reorganization of LOBs was SHRLEVEL NONE, with the following restrictions:

- ▶ No read/write availability during entire duration of the utility
- ▶ No space reclamation on the LOB table space
- ▶ No inline copy is created

As of DB2 9, you can run REORG TABLESPACE SHRLEVEL REFERENCE on an LOB table space, which provides better availability, space reclamation, and overall performance.

In DB2 10, REORG TABLESPACE SHRLEVEL CHANGE on an LOB table space is supported. This option provides even greater availability by enabling full read/write access during almost the entire duration of REORG.

### ***AUX column of base table for improved LOB handling***

Restrictions are removed that are related to the online reorganization of base table spaces that use LOB columns.

SHRLEVEL REFERENCE and CHANGE specifications are extended to add the keyword AUX YES/NO. This new keyword allows for the reorganization of LOB and XML table spaces that are associated with the base table. It is also possible to specify SHRLEVEL CHANGE for REORG of a LOB.

In DB2 9, when REORG is run against a PBG table space with LOBs, the data records from “part n” before REORG must go back into “part n” after REORG. Thus, there no data movement across partitions when a LOB is present, which can lead to REORG failure due to space issues.

DB2 10 addresses this issue with AUX YES support. Data movement across partitions during REORG on partition-by-growth with LOB is possible by using REORG to move the corresponding LOB data. The AUX keyword allows you to specify whether DB2 reorganizes associated auxiliary LOB table spaces along with the base table space. A value of YES means that LOB objects are handled automatically, and NO means that LOB objects are not touched during REORG.

### ***Multiple partition ranges can be specified in one REORG***

“Eliminate BUILD2 phase of online REORG” on page 232 gave an example of the advantage of specifying a range of partitions, 200-221, in one job in DB2 9. Only a subset of those partitions actually needed to be reorged. The others were unnecessary, but the advantage was that the NPIs were only reorged once. In DB2 10, it is now possible to specify in one job the partitions that you want to be reorged. Additionally, it only reorgs the NPIs one time, as shown in Example 19-12.

*Example 19-12 Partitions specified can be non-contiguous*

---

Partitions 200-204, 213, 218-221 are now allowed

---

### ***Reduce need for REORG INDEX with index leaf page list prefetch***

An index scan reads the leaf pages of the index in key sequence order. The DB2 sequential detection algorithm detects whether the leaf pages are physically organized well enough to kick off dynamic prefetch. If the leaf pages are disorganized, then DB2 9 runs synchronous I/Os. When doing sequential I/O, dynamic prefetch reads 128 KB per I/O, that is 32 pages if the page size is 4 KB. There is a significant incentive with DB2 9 to keep indexes organized. However, DB2 10 greatly mitigates any performance issues of a disorganized index. DB2 10 can use list prefetch I/O for the leaf pages. DB2 10 uses the N-1 level of the index to locate the leaf pages, whereas DB2 9 did not issue getpages for the non-leaf pages when doing an index scan. In a typical DB2 10 index scan, there are a few synchronous I/Os run when DB2 discovers that the index is disorganized. Thereafter, there is an occasional synchronous I/O for a non-leaf page, but the leaf pages are read using list prefetch. List prefetch on index leaf pages can greatly reduce the synchronous I/O waits for long running queries accessing disorganized indexes. Dynamic prefetch and list prefetch are also used for updating the nonpartitioning index (NPI), which happens to run an index scan.

There might be less need to run REORG INDEX because your application performance might be able to tolerate accessing data that is more disorganized than previous versions of DB2.

### ***Reduce need for REORG with compress on insert***

You can compress an index without using a dictionary. Compressing an index reduces the physical storage space that an index requires. By eliminating the need for a dictionary, index data can be compressed as soon as the first index entries are added to an index, and the space that would normally be used by a dictionary becomes available. You can choose whether you want to use index compression by specifying COMPRESS YES or COMPRESS NO on the CREATE INDEX and ALTER INDEX statements. It can become a pending ALTER if certain conditions are met. In addition, the buffer pool that is specified must be larger than a 4 K buffer pool.

### ***Create consistent image copies of multiple objects without outage***

Clients frequently need to produce data at specific points in time over a few to a few hundred DB2 objects. Often this data is brought to another system for reporting or testing. Before DB2 10, it was necessary to either run the DB2 QUIESCE utility or to find a naturally occurring “quiet time” that was a sensible business time (like the end of the day). It might be disruptive to run QUIESCE at the proper time. In DB2 10 with FlashCopy image copies, you can create consistent copies easily and without any outage at all. The copy can be produced with SHRLEVEL CHANGE.

When you create the copies, all the objects (partitions, table spaces, indexes) are provided in the job. You can use the utility control statement FCCOPYDDN FLASHCOPY CONSISTENT.

## CHECK increased availability

When you run the CHECK DATA or CHECK LOB utility on a DB2 subsystem of version 9 or less and the utility detects referential integrity violations, it places the entire table space into check pending (CHKP) restrictive state. As a result, the table space that was fully accessible before you ran the CHECK DATA or CHECK LOB utility is now completely restricted for any SQL access. This can be a major issue for some DB2 users who expect high availability in their DB2 operations. Usually if a table space has inconsistencies, they are limited to a small portion of the data.

In DB2 10, the utilities CHECK DATA and CHECK LOB have been changed so that they are more in line with the behavior that CHECK INDEX shows today. When you run all of these utilities with the new functionality turned on, DB2 no longer sets the CHKP status for table space or LOB objects when you run those utilities and DB2 finds inconsistencies. At first glance this might seem strange, but this process does not cause you any problems. Running the CHECK utility is a voluntary action.

Error handling is integrated into DB2 for dealing with inconsistencies in DB2 objects so that there is no need for the CHECK utility to make objects inaccessible. The main purpose of the CHECK utilities is to report problems and help you to obtain details about possible inconsistencies.

A DSNZPARM, called CHECK\_SETCHKP, is introduced by DB2 10 CM to allow you to turn this new function on or off:

- ▶ CHECK\_SETCHKP=NO means that DB2 should not set check pending state if inconsistencies are found.
- ▶ CHECK\_SETCHKP=YES tells DB2 to function as it did before DB2 10

## Add or update CDB without stopping DDF

You can make updates to the communications database (CDB) for new or existing entries. A CDB entry is only necessary when DB2 for z/OS is a requester. An existing entry where some of the information needs to be updated can now be done for TCP/IP in DB2 10. This process allows DDF to always be aware of the latest information in the CDB without a DDF recycle.

Updates to the following tables take place whenever a new or existing remote connection is requested:

- ▶ SYSIBM.LOCATIONS
- ▶ SYSIBM.IPNames
- ▶ SYSIBM.IPLIST

SNA users must restart DDF.

## Change DDF location aliases dynamically

You can add aliases dynamically to DDF by specifying the **MODIFY DDF** command and **ADD** options. You first define the alias:

```
-MODIFY DDF ALIAS (ITS0BLD8) ADD
```

You then issue **-MODIFY DDF ALIAS(ITS0BLD8)** with one configuration option, such as port number and so forth. Eventually you complete the task by issuing this command:

```
-MODIFY DDF ALIAS(ITS0BLD8) START
```

## 19.2.4 DB2 for z/OS 11

DB2 for z/OS 11 became generally available in October 2013. Table 19-5 shows the availability enhancements that it added.

For more detailed information about each of the listed enhancements, see *DB2 11 for z/OS Technical Overview*, SG24-8180.

Table 19-5 Availability enhancements for DB2 11

| Change                          | Described in                                                             | Notes (optional)                                                                                          |
|---------------------------------|--------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Basic object to Extended format | “Change DB2 objects from Basic format to Extended format” on page 242    | Optional.<br>Changes table space, partition, or index from 6-byte RBA/LRSN format to 10-byte format       |
| Extended object to Basic format | “Change the object from Extended format to Basic format” on page 246     | Optional.<br>Changes table space, partition, or index from 10-byte RBA/LRSN format to basic 6-byte format |
| DROP Column                     | “DROP column” on page 246                                                |                                                                                                           |
| Break in on persistent threads  | “Allow BIND / DDL / REORG to break in on persistent threads” on page 247 |                                                                                                           |
| Alter partition limit key       | “ALTER Partition Limit Key” on page 247                                  |                                                                                                           |
| Reorg Rebalance SHRLEVEL CHANGE | “REORG REBALANCE with SHRLEVEL CHANGE” on page 248                       |                                                                                                           |
| Reduce SWITCH phase of REORG    | “Reduce SWITCH phase of REORG” on page 248                               |                                                                                                           |

### Change DB2 objects from Basic format to Extended format

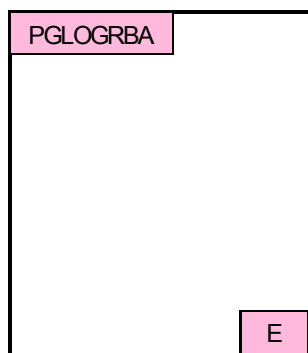
Some clients are close to or have reached the end of the 6-byte RBA. In DB2 11 new function mode (NFM), they want to convert to the 10-byte RBA or LRSN. It is a two-step process, and the steps can be run in any order:

1. Converting the BSDS data sets, as described in 19.1.4, “Convert BSDS to Extended RBA/LRSN in DB2 11” on page 227.
2. Changing the data page format in every object in your DB2 subsystem or data sharing group.

Figure 19-5 shows the format of a 4 K page in basic format, which is the page format that has been used since 1983. A brief explanation of the fields might be helpful:

- ▶ PGLOGRBA which is a 6-byte field that holds the RBA or LRSN of the last update to the page.
- ▶ The last part of a 4K page is a 2 byte block called PGTAIL. It is used by DB2 to check the page validity. It is either an “E” or an “N”.
- ▶ The second to the rightmost bit is called PGBigTailApplies. Notice that this bit is zero. Zeros mean PGBigTailApplies is off, meaning it does not apply and that this object is in basic format.

### ***Current 4K page format (Basic)***



- Pghead = PGLOGRBA (6 bytes stored)
- E is PGTAIL (2 bytes)
- Last byte is PGEND (parity)
  - “E” - x’11000101’ or
  - “N” - x’11010101’
  - Underlined bit called PGBigTailApplies
  - Zeros mean off (does not apply)
- So DB2 uses 6-byte RBA/LRSNs in pghead area

*Figure 19-5 4K page in basic format*

If the page is formatted this way, DB2 updates PGLOGRBA with the RBA or LRSN when the page is updated.

Figure 19-6 shows the format of a 4K page in extended format. It has these characteristics:

- ▶ The pghead field is the old PGLOGRBA and is not used in this format (though it might not be all zeros after conversion)
- ▶ PGBigTail is now 20 bytes. It is broken into several parts of which the most interesting is the 2-byte PGTAIL:
  - First byte is reserved.
  - Second byte is the same as the corresponding byte of the basic format page. Byte 2 has the next to rightmost bit contains “1”, which means PGBigTailApplies, which means the page is in extended format.
  - DB2 knows to write the RBA or LRSN in PGBigRBA (10 bytes) to the left of PGTAIL
  - Parity is checked using bits 2-4 of PGEND, either “100” or “101”.
  - The rest of the PGBigTail is reserved by IBM.

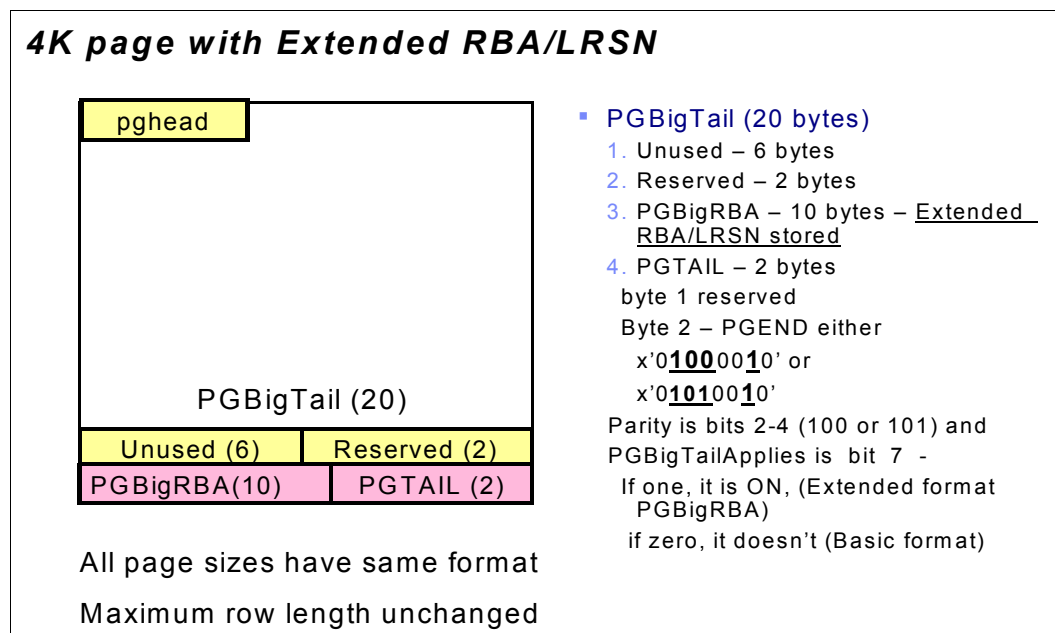


Figure 19-6 4K page in extended format

Other page types and sizes contain are formatted similarly. They are documented in *DB2 11 for z/OS Diagnosis Guide and Reference*, LY37-3222 (Chapter 21: “File Page Set formats”). This is a licensed manual. If you have DB2 11, you can find this manual in DSN1110.SDSNIVPD(DSNDR). Download it in binary format and it is a PDF.

**Attention:** The maximum number of rows has not changed in the extended format.

There is more space consumed by both table and indexes due to the PGBigTailApplies. Fewer rows might be able to fit in a data page or fewer keys might fit in an index page. Similarly, space map pages might cover a smaller range of objects. This means that you might have more space map pages following conversion to extended mode.

### How to convert

There is great flexibility in the conversion effort. The conversion itself is accomplished through a REORG with any SHRLEVEL. When you do it is entirely up to you. You can convert a table space, a partition, or an index at any time. Some partitions might be in extended format,

whereas others are still basic. Indexes can be in a different format from the underlying data. If you do not want to keep some of them in extended format, you can convert them back to basic by using another REORG. This flexibility is a reason the implementation of the extended RBA/LRSN capability is a major part of DB2 11. DB2 utilities and SQL and DDL can handle either page format.

You do not have to convert unless you are getting close to the end of the 6-byte RBA or LRSN. You will get plenty of warning messages with every active log switch, starting about a year before the end of the log is reached. But be aware that DB2 11 assumes all RBAs and LRSN values are 10 bytes, and performs conversion to 10 bytes temporarily every time a log record is accessed or written. You can set two DSNZPARM values and allow any object that is being reorged to also be converted to extended format when you get to NFM (they are ignored before that time with one exception). Those parameters are as follows:

- ▶ `OBJECT_CREATE_FORMAT`, which can be BASIC or EXTENDED. For newly created objects, this is the format that you want.
- ▶ `UTILITY_OBJECT_CONVERSION`, which can be BASIC, EXTENDED, NONE, or NOBASIC. This value concerns the format the object should be following REORG, LOAD, or REBUILD. BASIC and EXTENDED are self-explanatory:
  - NONE means for those utilities not to change the format on output.
  - NOBASIC means that the object is in EXTENDED format and changing it to BASIC is not allowed. The latter is useful when the 6-byte RBA/LRSN values have been exceeded by the subsystem or data sharing group. You do not want an inadvertent conversion to Basic format.

You can use utility control statements within the job to override the DSNZPARM values.

### ***Catalog and directory conversions***

Even before you are in NFM, you can begin conversion of parts of the DB2 catalog that are touched in ENFM. While the table spaces are being reorganized, it is efficient to them to Extended format at the same time. During ENFM, some of those objects are created, so both DSNZPARM, set during conversion mode (CM), ensures those catalog or directory table spaces are converted to extended format.

When you get to NFM, you can convert the catalog and directory using the SDSNSAMP job DSNTIJCV. Its assumption is that you are not likely to convert all those objects in one pass, so it is easy to stop it part way through, and continue it at another time. If you have already converted the ENFM catalog and directory objects, they are bypassed when DSNTIJCV runs.

You can recover an object in one mode back to a previous mode if you do not have a “log no” event, such as another REORG or LOAD. See Figure 19-7.

### **Recovery Considerations**

Example for table space ‘TS1’:

T1 - image copy  
T2 - quiesce  
T3 - convert to extended page format  
T4 - quiesce  
T5 - image copy  
T6 - quiesce

Q1: Can I recover TS1 back to time T2?

Ans: Yes. And the catalog is updated to indicate “basic” format

Q2: Can I then roll forward to time T4?

Ans: No. RECOVER does not allow log apply across a REORG

*Figure 19-7 Recovery to a previous mode*

## **Change the object from Extended format to Basic format**

“Change DB2 objects from Basic format to Extended format” on page 242 covers how to change objects, and you can easily reverse those actions. You can set the DSNZPARMs or use utility control statement overrides.

## **DROP column**

Up through DB2 10 there were these column options:

- ▶ Add column: A very old option
- ▶ Alter column characteristics: DB2 8
- ▶ DB2 9, described in “Rename a column” on page 230

Columns become obsolete over time. They take space in each row, in each image copy, and the DBA must “remember” that the column is obsolete. To eliminate the column used to be a long process. You had to unload data, drop the table, create a new table, and reload the table, all of which required an outage.

Now the client is able to drop the column easily with this command:

```
ALTER TABLE <table> DROP COLUMN <column_name> RESTRICT
```

It becomes a pending ALTER, and cannot be combined with any other ALTER. It is inserted into SYSIBM.SYSPENDING DDL. The restrictive state AREOR is set, providing access until you decide to REORG the table space.

Some (but not all) of the restrictions are listed, similar to those for column rename. A column cannot be dropped in these circumstances:

- ▶ The table space is not UTS
- ▶ The table is a created global temporary table, a system-period table, history table, or an MQT
- ▶ There are triggers defined on the table



- ▶ The column is referenced in a view
- ▶ It is an XML column, part of the table partitioning key, or an RI

You can drop the pending DDL, as always. After you REORG the table space, you cannot roll it back to before the REORG.

**Note:** Some pending ALTERS that have been materialized can be rolled back in DB2 11, but not this one.

## Allow BIND / DDL / REORG to break in on persistent threads

This capability provides significant availability.

Persistent threads are those bound with RELEASE DEALLOCATE. They have been used for many years for frequently accessed, performance sensitive applications with CICS and IMS. In DB2 10, with internal thread storage loaded above the 2 GB bar, DDF DBATs can also run with RELEASE DEALLOCATE. DB2 avoids allocating and deallocating the package on every commit. Fast column processing persists. Intent locks for table spaces and partitions do not have to be released and reacquired on every commit. All these enhancements yield CPU savings up to 20% compared to RELEASE (COMMIT) bind option.

The down side of persistent threads is that the package cannot be bound again until the thread ends, which might be days:

- ▶ BIND/REBIND cannot break in because the running thread holds shared (S) package locks and BIND/REBIND needs exclusive (X) package locks.
- ▶ DDL cannot break in because the running thread holds, in addition to the shared package locks, the intent locks, and DDL, need exclusive access.
- ▶ REORG needs exclusive access during its SWITCH phase to update the DBD.

DB2 11 allows these to break into RELEASE(DEALLOCATE) packages accessed by persistent threads, starting in CM. The package is processed as RELEASE (COMMIT) during this time. The package is allocated and deallocated, intent locks released and reacquired, and fast column processing is disabled during this time.

After these actions are completed, the package will continue to be accessed as RELEASE(DEALLOCATE).

The behavior is governed by a DSNZPARM PKGREL\_COMMIT. During commit process of the package, DB2 queries any X lock waiter for each RELEASE(DEALLOCATE) package with these exceptions:

- ▶ Package that is bound with KEEP\_DYNAMIC=YES
- ▶ Package has open and held cursor
- ▶ Commit issued within a stored procedure
- ▶ Thread is idle (thread must have active transaction and frequent commit)

The result of the break-in capability is that the application maintains availability during these events. This situation caused planned application outages in the past.

## ALTER Partition Limit Key

In previous versions of DB2, when you changed the limit key values, all affected partitions were placed in REORG-pending (REORG) status. The data were unavailable until you ran the REORG utility. That is still true for *index-controlled* partitioned table spaces.

In Version 11, when you change the limit key values for *table-controlled* partitioned table spaces, the data remains available and applications can continue to access the data. In most cases, the limit key changes are not materialized until the next time that you run REORG. The affected partitions are placed in advisory REORG-pending (AREOR) status. It becomes pending DDL.

**Important:** When converting index-controlled partitioning to table controlled, the easiest way to convert is to ALTER the clustering index to NOT CLUSTER and then ALTER it back to CLUSTER (the table remains table-controlled).

Then you can change the limit key in the same PENDINGDDL so that both actions occur with the same REORG activity

In a few cases, a change to a limit key value is immediately materialized, so there is no need to run REORG on the partitions. Immediate materialization occurs when DB2 determines that both of the following conditions are true:

- ▶ No data need to be moved between partitions.
- ▶ No other alter limit key operation is pending on the partition.

The alter limit key capability is limited to UTS PBR and spaces that are exclusively partitioned with table-controlled partitioning.

In addition to improved data availability, it is a pending ALTER that you can add to other pending definitional changes.

## REORG REBALANCE with SHRLEVEL CHANGE

DB2 Version 8 can rebalance partitions but only with SHRLEVEL REFERENCE, allowing only readers. DB2 11 permits full availability with SHRLEVEL CHANGE. In addition, it handles SCOPE PENDING.

REORG REBALANCE might fail if there is a non-unique partitioning key and data skew because the system cannot distribute data evenly across partitions. If there is not enough, but there still are parts to process, DB2 sets new limitkey values for empty partitions.

## Reduce SWITCH phase of REORG

Each version of DB2 tries to minimize the SWITCH phase of REORG, the only time there is total unavailability to the data. The SWITCH phase switches the shadow data sets to the actual data sets. The problem frequently occurs when a REORG is run on a subset of partitions, and usually involves a non-partitioning index (NPI) on that table space. The following situations could cause a REORG to fail:

1. REORG acquires drains on partitions serially and must wait behind claiming workload on affected partitions.
2. Transaction hold claims on partitions that are not being reorged, but also on the NPI of the table space, then later want a claim on one of the partitions being reorged, resulting in deadlock between REORG and the claiming transaction.

DB2 11 has two ways to minimize these problems:

1. DB2 sets a flag to prevent new CLAIM for the PARTs specified on REORG before starting the DRAIN for part 1. New application threads wait until REORG acquires all the DRAINS.
2. You can set the new DRAIN\_ALLPARTS YES option on the REORG (not the default). It tells DB2 to obtain the table space level drain on the entire partitioned table space temporarily first, then DRAIN the NPIs, release the drain on the entire partitioned table

space, and start draining the target data partitions and the indexes. This process provides relief by eliminating DRAIN timeout or deadlocks caused by the reverse order of object-draining by REORG and object-claiming by the Data Manipulation Language (DML) statement.





## IMS considerations

This chapter describes the various functions and features in the IMS product that are available to reduce or eliminate the need to shut down and restart an IMS image regularly. Before IMS Version 6, you had to shut down IMS twice a year to accommodate Daylight Saving Time. This chapter highlights some of the more recent changes to reduce planned outages. In addition, it offers some tips on how to handle those things that are not yet able to be changed dynamically.

Application availability for both planned and unplanned outages can be achieved to a high degree with the implementation of IBM Parallel Sysplex, including data sharing, shared queues, and IMS Connect. This sysplex architecture lets you apply IMS maintenance and even upgrade to a new IMS version, one IMS image at a time, without a planned outage of any application/business functions. Details on the implementation and use of IMS sysplex capabilities can be found in the IMS product manuals and several IBM Redbooks publications.

This chapter focuses primarily on two areas.

- ▶ Enhancements that provide more continuous availability of database data
- ▶ Enhancements that allow you to add, delete, or change IMS resource definitions without having to recycle IMS.

A third area includes tips on how you might deal with those items not yet addressed with a formal product solution. A Parallel Sysplex environment allows the recycling of IMS to be less disruptive. However, there are still operational issues to deal with, especially when there are connections to other subsystems such as CICS or DB2. Thus the ability to dynamically change parameters is generally less disruptive than recycling.

This chapter includes the following sections:

- ▶ IMS database considerations
- ▶ IMS Transaction Manager Considerations
- ▶ IMS system considerations
- ▶ Resources still requiring a recycle of IMS to change

## 20.1 IMS database considerations

One of the primary reasons to take a database offline is to perform database maintenance. As segments are added and deleted, many databases become disorganized. This might affect performance and, in an extreme case, cause applications to stop working when there is no usable space left in the database for inserting or replacing variable length segments.

Another reason to take a database offline, although typically less frequent, is to change certain attributes of the database or to make structural changes to the data.

This section provides ways to avoid those database outages or in some cases significantly shorten the time that they need to be unavailable.

### 20.1.1 Database reorganization

There are two primary reasons to perform reorganization. One is to optimize the placement of data to minimize I/O and regain performance. The other (less frequent) reason is to change the structure of the database records or the physical characteristics of the data set to accommodate application changes.

#### Fast Path DEDBs

Fast Path data entry databases (DEDBs) support full online reorganization with concurrent update capability. No outage is required unless you are doing restructuring. DEDBs were designed with online reorganization in mind.

With IMS Version 13, the DEDB alter function is added to allow more changes to be made to the database while remaining online. These changes include the following:

- ▶ The SIZE, UOW, or ROOT specifications of a DEDB area
- ▶ The randomizer that is used for the DEDB area
- ▶ Increase or decrease the Root Addressable Area or Independent Overflow

The DEDB alter function does not support DEDB databases that are in Virtual Storage Option (VSO) or Shared Virtual Storage Option (SVSO) mode. However, DEDBs can be removed from VSO or SVSO mode with the **/VUNLOAD** command before running the DEDB Alter utility.

To use the Alter function, you must use a two stage DEDB randomizer, which you might already be using.

For more information about DEDB online reorganization and the DEDB Alter function, see *IMS Version 13 Release Planning*, GC19-3658, and the product manuals.

#### High availability large database (HALDB)

HALDBs were introduced back in IMS Version 7. By placing data in multiple partitions, they can be reorganized concurrently to reduce the database outage time. With IMS Version 9, the Online Reorganization function was introduced to allow full online update capability while reorganizing a single partition or multiple partitions. HALDB Online Reorg eliminates database outages for most reorganizations.

With IMS Version 13, the HALDB alter function is available. When combined with Online Reorg, it helps make the following structural changes to PHDAM and PHIDAM databases in addition to the standard online reorganization function:

- ▶ Add a field at the end of a segment
- ▶ Increase the length of a segment
- ▶ Define new fields that redefine existing fields and space in the segment

The HALDB alter process involves the following steps:

1. Define the structural changes and generate a new DBD that goes into the DBD library.
2. Run the ACB Maintenance utility, which creates a new ACB member in the staging ACB library.
3. Issue the type-2 **INITIATE OLREORG** command, with the new **ALTER** option, to start the HALDB alter process. IMS reads the staging ACB library and applies the changes to the online database.
4. After the HALDB alter process finishes successfully, issue online change commands to activate the new ACB member.
5. After the new ACB members are active, the new database structure can be used.

### **IMS Full Function database (non-HALDB)**

Full Function IMS databases do not yet have the online reorganization capability and still require the database to be taken offline for some time. However, there is a separately priced tool called Online Reorganization Facility that reduces the outage time to a small window at the end of the reorganization process. If a short database outage is acceptable, this tool might meet your needs.

For those databases that require total availability, consider the benefits of changing to HALDB. These include online reorganization and alter, even if only a single partition is necessary. An alternative to HALDB is DEDBs. These not only have online reorganization and alter functions, but also support Multiple Area Data Sets for high availability.

### **Database versioning**

Database versioning allows you to assign user-defined version identifiers to different versions of the structure of a database. These identifiers enable you to make structural changes to a database while providing multiple views of physical data to various applications.

New applications that reference a newer structure of a database can be brought online without affecting applications that use previous database structures.

Unchanged applications, which do not have to be sensitive to the new physical structure of the database, can continue to access the database.

Database versioning can be used for the following types of databases:

- ▶ DEDB
- ▶ HDAM
- ▶ HIDAM
- ▶ PHDAM
- ▶ PHIDAM

The database versioning function can be used with the database alter functions to track different versions of the structure of a database.

Database versioning supports the following structural changes to a database:

- ▶ Increasing the length of a segment
- ▶ Adding a field at the end of a segment

After a DBD has a version number, applications can use the INIT VERSION DL/I call to access a specific version of the database or specify the database version for an application program on the DBVER parm in the PCB statement. A default VERSION is specified in the DFSDFxxx member of IMS proclib.

## 20.1.2 Dynamic database buffering

This function of IMS is technically considered part of the IMS systems area. However, because this function affects database availability, it is included in the database section.

### Dynamic Fast Path buffers

IMS Version 11 gave Fast Path users the option to replace the traditional Fast Path buffer manager with a new system-managed FP buffer pool.

If you use the Fast Path 64-bit buffer manager, you do not have to specify the DFSPBxxx parameters DBBF, DBFX, and BSIZ. Those parameters were difficult to understand and properly manage without using the trial and error method. With the Fast Path 64-bit buffer manager enabled, Fast Path automatically allocates buffer subpools up to the maximum size specified by a parameter you provide.

In particular, the Fast Path 64-bit buffer manager allocates subpools for different DEDB CI sizes, rather than insisting on a single CI size. This approach means that you might consider retuning your DEDBs to more efficient CI sizes without affecting other DEDBs or stopping IMS.

If you add a Fast Path database through online change, and this database cannot fit into an existing subpool, the Fast Path 64-bit buffer manager allocates a fresh subpool of a suitable size. Previously, this process required an IMS outage.

Specify 64-bit Fast Path buffer manager (FPBP64=Y) to take advantage of this enhancement. With FPBP64=Y specified IMS ignores any DBBF, DBFX, and BSIZ specifications.

### Dynamic change capability for Full Function database buffers

With IMS Version 12, users can add, change, and delete full function buffer pools. This support is provided using new specifications in the DFSVSMxx proclib member with the **UPDATE POOL** command. With this support, full function buffer pools can be managed without restarting IMS.

IMS internally quiesces application read and update activity so that the UPDATE POOL command can complete with little disruption to transaction workloads.

In earlier IMS versions, the Virtual Storage Access Method (VSAM) and Overflow Sequential Access Method (OSAM) buffer pool definitions were only stored in the DFSVSMxx proclib member. This member is loaded only one time during IMS initialization. No facility is available to change the buffer pool definitions without first changing the DFSVSMxx member in proclib and then restarting IMS.

IMS version 12 offers this new feature for dynamically adding, updating, and deleting VSAM and OSAM buffer pools. The initial VSAM and OSAM buffer pool specifications still exist in the DFSVSMxx proclib member, and are loaded during normal restart. However, new VSAM and OSAM buffer pools can be added and existing buffer pools can be changed by using



specifications in one or more DFSDFxxx proclib members with the type-2 **UPDATE POOL** command.

This support allows changing full function pools to meet workload changes and accommodate new database block sizes. The result is that in most cases no outage is necessary to change the Full Function buffer pools.

### 20.1.3 Miscellaneous database considerations

This section covers other reasons a database might need to be unavailable, and possible ways to minimize or eliminate database outages.

#### CA reclaim

When IMS database KSDS records are erased with z/OS 1.11 and previous releases, VSAM CI reclaim does not reclaim the empty CI that has the highest key of the control area (CA). This otherwise empty CA occupies the index structure as though it was not empty. If an application reinserts records with the erased keys or keys of nearby values, those empty CAs are reused. However, if the application erases a range of keys and does not reuse those keys or only inserts records with ever-higher keys, VSAM does not reclaim or reuse those empty CAs with lower keys. The failure to reclaim the CAs results in wasted disk space and can cause performance problems in the index search process, making it necessary to reorganize the index. z/OS 1.12 added a function to reclaim these empty CIs so they can be reused, and IMS Version 11 and later added support for this function that allows these databases to remain online.

In addition to databases, the CA reclaim enhancement can also benefit the DBRC Recon data sets by reducing or eliminating the need for reorganization.

#### Database quiesce

Database recovery points must be established in the unlikely event a database must be recovered from some failure. Before IMS Version 11, this was typically accomplished by taking the database offline with a **/DBR** command. This means the database is unavailable for some time.

With IMS Version 11 and the new **QUIESCE** option on the **UPDATE DB** command, a recovery point can be created while the database remains available. Read processing continues, but there might be a slight delay with updates while work is temporarily quiesced.

The QUIESCE keyword has two options:

- ▶ QUIESCE and GO tell IMS to resume all activity as soon as the recovery point is established.
- ▶ QUIESCE and HOLD tell IMS to wait for the STOP(QUIESCE) to resume update activity. This option is typically used when an image copy is being taken.

#### Database Recovery Facility

DRF provides a point-in-time recovery capability that eliminates the need to take recovery points as described above. It can be used to recover databases to any time that you choose. The databases remain fully available. This is a separately priced tool, and not part of the base IMS product.

#### Database Image Copy 2

IMS Version 10 enhanced the Database Image Copy 2 utility (DFSUDMT0), the Database Recovery utility (DFSURDB0), and Database Recovery Control (DBRC) to support image

copies that are taken using the DFSMS fast replication copy function. The fast replication option of the Database Image Copy 2 utility creates image copies quickly. You can create clean or concurrent fast replication image copies and register them with DBRC for use in recovery. Fast replication image copies are exact copies of the input data sets, and are not formatted like image copies taken by the Database Image Copy 2 utility using the DFSMS concurrent copy option.

Use of the fast replication image copy combined with the QUIESCE and HOLD capability described previously can allow clean image copies to be taken with little or no impact to the user.

## 20.2 IMS Transaction Manager Considerations

IMS Transaction Manager has the Extended Terminal Option (ETO) function to support dynamic Systems Network Architecture (SNA) terminal sessions without the need to sysgen and recycle the IMS subsystem. TCP/IP connections are also dynamic. The TCP/IP connection is generally to IMS Connect, which then passes the message to IMS Open Transaction Manager Access (OTMA), which in turn creates the control blocks for this connection dynamically. This process leaves Multiple Systems Coupling (MSC) links still requiring static definitions, but with some planning these can also be made reasonably dynamic.

### 20.2.1 Dynamic change for OTMA routing descriptors

Before IMS Version 11, if you wanted to change the destination routing descriptors, you had to wait for a scheduled outage or for IMS to be restarted. With IMS Version 11 and later, you can use type-2 commands for OTMA destination routing descriptors to change the descriptors dynamically, removing the need to recycle IMS.

You can dynamically add, update, delete, and query destination routing descriptors by using these commands: **CREATE OTMADESC**, **UPDATE OTMADESC**, **DELETE OTMADESC**, and **QUERY OTMADESC**. This support also removes the need for these descriptors to be in any specific order.

### 20.2.2 Dynamically change the IMS Connect configuration

IMS Version 13 provides new IMS commands to dynamically change certain IMS Connect configuration settings. **CREATE IMSCON TYPE(PORT)** is equivalent to the PORT substatement in the TCPIP statement and the DRDAPORT substatement in the ODACCESS statement of the ICON config.

The **CREATE IMSCON TYPE(DATASTORE)** command is equivalent to creating the DATASTORE statement in the HWSCFGxx member of IMS Proclib. When the command completes, IMS Connect starts the IMS data store and sets the IMS data store status to CONNECT if it is successful and DISCONNECT if it is not successful.

The **CREATE IMSCON TYPE(PORT)** command is equivalent to creating the PORT substatement of the TCPIP statement of the HWSCFGxx member of IMS Proclib. When the command completes, IMS Connect starts the port and sets the port status to ACTIVE if it is successful and NOT ACTIVE if it is not successful.

There is no longer a need to recycle IMS Connect for these changes.

### 20.2.3 MSC dynamic reconfiguration

Although Multiple Systems Coupling (MSC) links must still be statically defined, it is possible to have a reasonably dynamic MSC configuration. With proper planning there are ways to modify, expand, and manage your MSC network without requiring an outage.

IMS Version 11 added support for VTAM Generic Resource (GRSNAME). This allows you to move sessions within the IMSplex without affecting the remote IMSs.

By defining dummy links in your IMS static system definition, you can add logical sessions to existing connections or even create a connection to a new IMS. You can do this by dynamically modifying and activating the dummy link by using IMS commands.

IMS Version 12 added enhancements to the **UPDATE** command so that you can change almost any MSC configuration characteristic you want.

**UPDATE MSLINK**, **UPDATE MSPLINK**, and **UPDATE MSNAME** commands can do everything the traditional **/MSASSIGN LINK** and **/CHANGE LINK** commands can do and much more:

- ▶ **UPDATE MSPLINK** can change automatic session restart (ASR), msplinkname, rmtims, genlogon, modetablename, and VTAM\_node\_name.
- ▶ **UPDATE MSLINK** can change linkname and partner\_id.
- ▶ **UPDATE MSNAME** can change the remote and local SIDs.

For a complete description of the **UPDATE** command and the things it can do, see *IMS Version 13 Operations and Automation*, SC19-3657.

If you do not have dummy links defined and more capacity is needed, you can activate a function called BANDWIDTH mode. This function might give you enough increased throughput to avoid a planned outage to define new logical links.

IMS Version 10 introduced BANDWIDTH mode, a new logical link mode, which allows you to control the bandwidth of logical links to dynamically manage the capacity of your MSC network. When bandwidth mode is off, MSC sends one message per network send. When bandwidth mode is on, IMS consolidates messages in buffers and MSC sends and receives multiple messages in the same transmission. Larger buffers allow more messages to be sent together in the same send.

Using the **UPDATE MSLINK** command, you can dynamically turn BANDWIDTH on and off, and dynamically increase or decrease the size of the link buffers as well. Bandwidth mode might also reduce synchronous I/O by requiring fewer check writes on the sending side, resulting in fewer delays.

IMS Version 12 adds support for MSC communications between IMSs using TCP/IP. If you define both TCP/IP and VTAM physical links between the same IMSs, you have an effective back-up configuration. If either VTAM or TCP/IP is not available (planned or unplanned), your MSC traffic is not affected. For example, if the VTAM network must be taken down, you can assign (**/MSASSIGN**) the VTAM logical links from their associated VTAM physical link to the TCP/IP physical link, and restart (**/RSTART**) them.

## 20.3 IMS system considerations

This section deals with the IMS systems area. It generally applies to all IMS environments such as CICS/DBCTL and IMS TM with IMS DB or DB2.

### 20.3.1 Member Online Change (MOLC)

Although Online Change has been available for many years and eliminates the need to recycle IMS to change program (PSB) and database (DBD) definitions, there were situations where it was easier to recycle IMS because resources in use by running programs cannot easily be quiesced. IMS Version 10 provides two enhancements for online change processing: Member Online Change (MOLC) and enhanced COMMIT.

MOLC is an optional alternative to a complete refresh of ACBLIB. MOLC enables you to add or change individual members of the ACB library. You can bring these new or changed members online without having to switch the entire active and inactive libraries. Only resources that are directly affected by the change are quiesced, which reduces the impact on work in progress. Changed members are copied from the staging ACBLIB directly to the active ACBLIB. MOLC is started by specifying **TYPE(ACBMBR)** on the **INITIATE OLC** command. An OLCSTAT data set and a CSL environment are required.

IMS Version 10 also enhanced the COMMIT phase for all types of online change to improve the chances for success. Commits no longer fail because of a transaction in progress if that transaction is not directly affected by the online change. Changes made to an APPLCTN or DATABASE macro do not directly affect a transaction if the TRANSACT macro is not changed. The commit enhancement applies to both local (/MODIFY) and global (INIT OLC) online change.

### 20.3.2 ACBLIB dynamic allocation

When ACBLIBs are allocated by JCL, they cannot be moved, resized, or compressed while IMS is active. IMS Version 11 provides the option to use dynamic allocation (DFSMDA) instead of JCL. With dynamic allocation, only the active ACBLIB is allocated. The inactive ACBLIB can be moved, resized, or compressed, and then made active with an online change. Dynamic allocation also allows you to add more ACBLIB data sets to the concatenation.

### 20.3.3 ACBIN64

IMS Version 11 provides an optional 64-bit storage pool to cache ACBs for non-resident PSBs and DMBs. Although this is primarily a performance-related option to avoid having to do I/O to read blocks from ACBLIB, it might also delay or avoid planned outages just to change the size of the CSA or DLI PSB pools.

IMS must maintain a copy of each concurrently scheduled PSB in the 31-bit PSB pool. Trying to balance the size of the PSB pool with other pools and control blocks while keeping I/O to a minimum can be difficult. With 64-bit storage for ACBLIB, IMS can cache all the PSBs and DMBs such that the blocks will only ever be read from DASD the first time they are referenced and then cached in the 64-bit pool. This means that the 31-bit PSB pools do not need to be large enough to hold all the PSBs times the number of regions capable of parallel scheduling them if it is large enough to handle the concurrently scheduled application programs because I/O is no longer necessary.

**Note:** The 31-bit DMB pool must be large enough to hold all DMBs for all databases that can be opened concurrently. A significant performance penalty could otherwise be seen.

### 20.3.4 Language Environment dynamic runtime options

Before IMS Version 8, if you wanted to modify Language Environment options for an IMS application program, it was necessary to relink the IMS application program with a new version of the CEEUOPT module or the CEEBXITA user exit and recycle dependent regions to bring the modified program online.

IMS Version 8 provides a way to dynamically change Language Environment runtime options without relinking programs and recycling dependent regions. You replace the LE CEEBXITA exit with the IMS-provided DFSBXITA exit. The DFSBXITA exit can either be linked into your application program as CEEBXITA, or it can be linked into a private Language Environment library for use by any application program in an IMS dependent region that includes the private library.

After you have DFSBXITA in place, you can use the **UPDATE LE** command to make dynamic changes to the Language Environment runtime options, including debug, storage, addressing mode, and environment reusability attributes. **UPDATE LE** allows you to specify particular application instances to which the options should apply. You can filter by transaction code, LTERM, user ID, and program name.

### 20.3.5 Refreshable user exits

IMS Version 13 extends the IMS User Exit Refresh function added in IMS Version 11 and enhanced in IMS Version 12. Users can now dynamically refresh the modules for certain exit types with the new **REFRESH USEREXIT** command. IMS does not have to be stopped and started to recognize the changed or new exit routines. User exit enhancements also include the new **QUERY USEREXIT** command to query information about exit routines.

The **QUERY** and **REFRESH** commands only apply to the exits defined in the **USER\_EXITS** section of the **DFSDFxxx** member.

As mentioned earlier, the **REFRESH** command support was added to IMS Version 11 and expanded in Version 12. IMS Version 13 expands the list considerably to support all of the following user exits:

- ▶ BSEX - DFSBSEX0 (Build Security Environment Exit)
- ▶ PPUE - DFSPUE0 (Partner Product User Exit)
- ▶ RESTART - no specific name but DFSRSTX0 sample provided (Restart Exit)
- ▶ INITTERM - no specific name (Initialization and Termination User Exit)
- ▶ ICQSEVNT - no specific name (CQS Event Exit)
- ▶ ICQSSTEV - no specific name (CQS Structure Event Exit)
- ▶ LOGEDIT - DFSFLGE0 (Log Edit Exit)
- ▶ LOGWRT - DFSFLGX0 - (Log Write Exit)
- ▶ NDMX - DFSNDMX0 (Non-Discardable Message Exit)
- ▶ RASE - DFSRAS00 (Resource Access Security Exit)
- ▶ OTMAIOED - DFSYIOE0 (OTMA Input/Output Edit Exit)
- ▶ OTMARTUX - DFSYRTUX (OTMA Resume Tpipe Security Exit)
- ▶ OTMAYPRX - DFSYPRX0 (OTMA Destination Resolution Exit)

### 20.3.6 Dynamic LOCKTIME

The **LOCKTIME** function has been available since IMS Version 9 to provide a way of preventing applications from waiting too long for a database lock. It also allows the application to return a meaningful message back to the user such as 'system busy, try again later'.

IMS Version 10 provided a new parameter to the **UPDATE** command to allow changing the **LOCKTIME** value dynamically. Before this enhancement, IMS had to be recycled to change the **LOCKTIME** value.

A periodic application (for instance, one that runs once a quarter) with many database updates might require a different setting for **LOCKTIME** to avoid application timeouts. This enhancement allows the user to respond to changing business conditions without an outage.

A parameter is added to the **UPDATE** type-2 command to dynamically modify the **LOCKTIME** parameter. Another parameter is added to the **QUERY** type-2 command that returns that current setting of **LOCKTIME**.

### 20.3.7 Dynamic resource definition (DRD)

In IMS Version 10, dynamic resource definition (DRD) became available. DRD allows you to dynamically create, update, delete, and query your resource definitions.

When you use DRD to manage your resource definitions while IMS is online, the requirement for a **MODBLKS** **SYSGEN** is eliminated. You also no longer need to recycle your IMS system or run a **MODBLKS** online change to bring these newly defined resources online.

DRD in IMS Version 10 uses a **BSAM** data set called the system resource definition data set (**RDDS**) to hold your stored resource definitions. There are a minimum of two, but generally have three **RDDS**s that are unique to each IMS image.

In IMS Version 12, users of IMS DRD can choose to implement the new repository function to manage **MODBLKS** resources in a single centralized location. From these centralized data sets, the resource definitions can be stored and retrieved by different IMS systems that are together in an **IMSplex**. The data sets can be shared among the systems and be dynamically updated.

If you are already using DRD with separate **RDDS** for each IMS system, you can complete a few simple steps to begin using the repository instead. This eliminates the need to manually coordinate and manage several **RDDS**s across different IMS systems. If you have never implemented DRD, you can use the definitions that exist in your **MODBLKS** data set as a starting point, eventually porting them to the new repository.

DRD improves IMS availability by allowing dynamic creation, deletion, and updating of resource definitions. It reduces the requirements for planned outages and eliminates unavailability associated with **MODBLKS** **OLC**.

### 20.3.8 DBRC

As mentioned previously most installations had to shut down IMS twice a year when the time changed. That requirement went away in IMS Version 6 when all internal times were changed to use Coordinated Universal Time.

The other main cause of having to shut down DBRC was due to the ever-growing **PRILOG** record, which could ultimately grow to the maximum **VSAM** recordsize. The only way to deal with this was to shut down and restart DBRC, which meant an IMS outage. IMS Version 8 added two enhancements to deal with this issue. One was to increase the maximum recordsize to 16 megabytes. This change also allowed the **VSAM** recordsize to be reduced and also removed the need for **VSAM** spanned records because DBRC used its own internal method to handle large records. The other enhancement was to run automatic **PRILOG** record compression, which can reduce the size of the record.

DBRC has also supported the online **UPGRADE** command to facilitate upgrading the RECONS to a new IMS version and allow coexistence between multiple versions of IMS (within two versions) without ever taking down IMS.

Although these changes were implemented many years ago, they continue to be significant.

### **20.3.9 IMS Version upgrades without an IPL**

As of IMS Version 9, the IMS Type 2 SVC and the IMS Type 4 SVC modules can both be dynamically updated without requiring an IPL to activate. For the Type 4, even IMS can remain active without the need to recycle. However, keep in mind that both the T2 and T4 SVCs should still be bound into SYS1.NUCLEUS (T2) or LPALIB/MLPALIB (T4) to prevent a z/OS IPL from regressing these modules.

With IMS Version 9, the IMS resource cleanup module (DFSMRCL0) is also no longer required to be bound into LPA/MLPA. In addition, the zap to CSECT IEAVTRML of module IGC0001C is no longer necessary. The first instance of IMS to be started after an IPL dynamically installs the IMS resource manager. This copy persists across IMS restarts. However, if a newer version of IMS is brought up, the older resource manager is replaced, so there is no need to IPL to start a newer IMS version.

With IMS Version 12, the MSC CTC channel end appendage DFSCMC10 exists only in the IMS RESLIB and is loaded during IMS initialization. The IGG019xx name of this module is no longer generated and no link into LPA/MLPA is required, thus removing any need to IPL the z/OS image.

## **20.4 Resources still requiring a recycle of IMS to change**

This section addresses some of the known items that, at the time of this writing, still require IMS to be stopped and restarted. Most of these do not require any system definition changes and can be implemented with a change to the startup parms. In many cases, planning ahead and selecting the appropriate settings can eliminate or avoid any planned outages for extended periods of time.

### **20.4.1 Variable storage pools**

Variable pool storage does not mean that the size of the pool is variable, but rather the blocks that are brought into the pool are variable in size.

These pools include the DLI and ECSA PSB pools, the PSB work pool, the DMB pool, the database work pool, and the Fast Path EPCB pool.

The use of ACBIN64 might allow the same or smaller size PSB pools to be used without a significant impact on performance if the pools are large enough to hold the largest PSB times the number of dependent regions.

Continue to size the DMB pool to hold all the possible open databases to avoid the delays that are associated with closing and opening database data sets.

The PSBW pool must also be sized to hold the largest workspace requirement times the number of regions.

DMB work pool requirements have always been small, so fine-tuning has generally never been an issue.

## 20.4.2 Fixed size (SPM) storage pools

These pools generally do not require changing because they are designed to expand and compress as needed. However, performance is sometimes enhanced by overriding the default specifications using the DFSSPMxx member of IMS Proclib. If you override the default numbers, consider keeping multiple sizes, especially larger sizes than you normally need. This might save the need to recycle when something changes in your workload and a larger buffer size is needed.

## 20.4.3 Log buffers

IMS requires a recycle to change the number of log buffers. However, with IMS Version 12 the log buffers might be allocated in 64-bit virtual storage which, depending on the number specified, can free considerable ECSA storage. This storage can be used for other pools, potentially allowing them to be allocated large enough to prevent a recycle to change them.

Of course you can also allocate many more buffers to avoid buffer waits that can cause poor performance. By allocating more than you normally need, you might never need to change the value.

**Note:** IMS Version 13 has some internal enhancements to optimize the handling of large numbers of log buffers. Even though Version 12 allows large numbers of buffers, it is typically best to keep the number below 2,000 until you are using Version 13.

Log buffers are obtained in 64-bit storage when all the following are true:

- ▶ The new BUFSTOR parameter of the DFSVSMxx PROCLIB member specifies BUFSTOR=64.
- ▶ The OLDS block size is a multiple of 4096.
- ▶ The OLDS are on extended format data sets.





## WebSphere MQ for z/OS

This chapter describes some of the facilities that are currently available to help avoid planned subsystem or system outages from WebSphere MQ for z/OS. However, it is understood that planned outages are ultimately necessary for subsystem and system maintenance. For such situations, this chapter provides remedies available in WebSphere MQ for z/OS to help reduce or remove the impact of the downtime due to these planned outages on application availability. Finally, it lists changes that do require a subsystem restart or a system IPL.

For a more extensive description of many of the topics described here, see the WebSphere MQ Version 7.1 Information Center at:

<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/index.jsp>

**Note:** This chapter assumes WebSphere MQ for z/OS. Where a particular feature or enhancement is relevant only in a certain version or release of WebSphere MQ for z/OS, reference will be made to the version and release in question. It is also assumed that the installation under consideration is WebSphere MQ for z/OS Version 7.0.0 or higher.

This chapter includes the following sections:

- ▶ Basic concepts and terms
- ▶ Availability and planned outages
- ▶ Applying maintenance during subsystem operation
- ▶ Changes that can be made dynamically
- ▶ Changes requiring a queue manager restart or IPL
- ▶ Error recovery (without a system or subsystem restart)

## 21.1 Basic concepts and terms

This chapter refers to the WebSphere MQ for z/OS implementation of the queue manager, queue, and message, simply as queue manager, queue, and message.

### 21.1.1 Messages / message queues

There are two types of WebSphere MQ messages. A queue can contain a mixture of either of them. However, to distinguish them, terms associated with logging must be defined:

- ▶ The *queue manager logs* are data sets for recording changes to message data to provide transaction recovery. This is often referred to as *logging*, while a message is *logged* if it is written to the log. The logs are also used by the queue manager to record changes to queue manager objects such as private queues.
- ▶ *Messages* are collections of binary or character data that are processed by applications.
- ▶ *Message queues* (or *queues*) are objects that store messages.
- ▶ A *queue manager* provides an implementation of message queues and is responsible for managing message queues, storing and providing access to message data, and transferring messages to other queue managers and applications.
- ▶ *Messaging applications or programs* can connect to a queue manager to access messages, such as put and get messages from queues.
- ▶ *Persistent messages* are logged and written to the queue data files. If a queue manager is started again after a failure, it can recover persistent messages from the logs, if it was unable to write them to the queue. Messages that are persistent are also retained on the queues over restarts.
- ▶ *Nonpersistent messages* are not logged and are discarded if a queue manager stops, whether it is stopped as a result of an operator command or due to a failure. Nonpersistent messages that are stored on shared queues are an exception to this. They persist if the Parallel Coupling Facility remains available.

### 21.1.2 Queues

The following are terms used when discussing queues:

- ▶ A *local queue* describes a queue in respect of an application that can be directly accessed by the queue manager to which the program is connected. A queue manager provides the following implementations of a local queue.
- ▶ A *private queue* is a queue whose storage is backed by a type of VSAM data set called a *page set*, which can be accessed only by the hosting queue manager.
- ▶ On a *shared queue*, messages can be accessed by one or more queue managers that are in a Parallel Sysplex. The queue managers that can access the same set of shared queues form a group called a *queue-sharing group*.

### 21.1.3 Unit of work

A *unit of work* (UOW) is a logical collection of serialized transactions. When a program puts messages to queues within a UOW, those messages are made visible to other programs only when the program commits the UOW. To commit a UOW, all updates must be successful to preserve data integrity. A local UOW only involves transactions with messages on a single

queue manager. A global UOW can involve updates to other resources, such as database objects like DB2 tables.

## 21.2 Availability and planned outages

Although a description of the availability of a queue manager subsystem often relates to configuration of failover and recovery, perhaps a more important aspect is to ensure continued availability of message processing to applications. For this reason, this chapter focuses on configurations and features of WebSphere MQ for z/OS that help minimize or make unnecessary either planned subsystem or system outages, and those that provide greater uninterrupted application availability.

### 21.2.1 Minimizing a queue manager outage due to an image IPL

Sometimes it is necessary to IPL a z/OS image. In such situations, the queue managers running on these systems must be stopped. To reduce outage time for queue manager message processing, it might be desirable to be able to start the queue manager on another z/OS image.

#### Moving queue managers between z/OS systems

You need the means of creating an environment in which queue managers can be started on alternative systems to provide greater application availability. To do this, you need these prerequisites:

- ▶ Ensure that the queue manager data sets, such as the page sets and logs, are defined on shared DASD volumes within the sysplex.
- ▶ The subsystem also needs to be defined with a unique name within the sysplex and with a sysplex-wide scope.
- ▶ Each z/OS image needs to be maintained at the same queue manager Early code level.
- ▶ Additionally, the TCP virtual IP addresses should be available on each TCP stack in the sysplex.

For an explanation as to what the Early code is, see 21.5.1, “Does an early code upgrade need a queue manager restart or IPL?” on page 273.

In this configuration, queue managers are no longer tied to a particular z/OS image, which allows the queue manager to be restarted on a different z/OS image within the sysplex. This configuration offers not only a failover environment, but also a means of allowing for planned system outages while reducing impact to application availability. In such situations, the queue manager can be quickly restarted on an unaffected image. A client can use this facility by planning outages during quieter periods of activities for the subsystem in question.

### 21.2.2 Availability of messaging in a QSG during an image IPL

By configuring queue managers in a queue-sharing group (QSG) within a *Parallel Sysplex* (a Sysplex together with a *coupling facility* (CF)), messages can continue to be made available to applications without failing over a queue manager to another system during an IPL. To achieve this, message queues are defined as *shared queues*. Shared queues are queues that are held in a CF. Use of a CF increases the capacity for data sharing among sysplex systems by the addition of dedicated CF memory called *Structures* that can be accessed throughout

the sysplex. Queue managers in a QSG use CF *List Structures* to hold shared queues and QSG status information.

As the CF is available throughout the sysplex, shared queues can be accessed by multiple queue managers running on different LPARs, providing higher message availability and workload balancing.

In particular, a queue manager in a QSG might be stopped and message processing can be failed over to another queue manager in the QSG that also has access to the same queue. The alternate queue manager might be on the same system or another system within the Parallel Sysplex. That means that if one of the queue managers is not available within the QSG to access a particular shared queue, any other queue manager in the QSG can access the same queue. In such a setup, an application need no longer depend on the availability of a particular queue manager within the QSG to be able to put and get messages.

Programs that connect to WebSphere MQ from the same system image, such as CICS, IMS or batch applications, depending on business logic, might be developed to process messages from a shared queue from multiple images each running a queue manager in the same QSG and a copy of the corresponding program. Applications that connect to WebSphere MQ for z/OS remotely (*WebSphere MQ Clients*) need to connect only to the QSG by specifying the QSG name on the connection, and need not be bound to a particular queue manager or image within the sysplex.

### **21.2.3 Can a WebSphere MQ channel be automatically rerouted if a queue manager is made unavailable?**

The appropriate configuration of your queue managers within a QSG with the use of shared channels can allow WebSphere MQ for z/OS to be made highly available within a network.

Planned outages of systems or subsystems can effectively be hidden from the network by configuring channel initiators to use generic ports for inbound attach requests. Any such request can be routed to any available channel initiator in the QSG configured in this way.

Shared outbound channels process messages that they send from a shared transmission queue. Information about the status of a shared channel is held in one place and available to the QSG at large. By using queue manager peer recovery within a QSG, a channel can be restarted automatically on a different channel initiator in the QSG if the channel initiator or corresponding queue manager is stopped. That means that the messages are available to each queue manager in the QSG. This allows for continued outbound traffic during planned outages.

### **21.2.4 Can a queue remain available if a queue manager is stopped and you are not using shared queues?**

Clusters provide increased availability and workload balancing. A cluster is a network of queue managers that are logically associated in some way. The queue managers in a cluster might be physically remote. For example, they might represent the branches of an international chain store and be located in different countries. Each cluster within an enterprise must have a unique name. More information about clusters can be found in *Queue Manager Clusters*, SC34-6589.

- Private queues can be made available to a WebSphere MQ cluster. Any available queue manager in a cluster can route a message to any other queue in the same cluster hosted by an available queue manager in the cluster. In addition, more than one instance of the

cluster queue can be used in an application environment that does not rely on message affinities.

- ▶ You can also group queue managers into a cluster. Queue managers in a cluster can make the queues that they host available to every other queue manager in the same cluster. Queue managers in a cluster can communicate directly with each other over a network, without the need for manually defining a transmission queue, channel, and remote queue. Each queue manager in the cluster has a single transmission queue (the `SYSTEM.CLUSTER.TRANSMIT.QUEUE`) that is used to transmit messages to any queue in the cluster hosted on any other queue manager in the cluster.
- ▶ You can increase the number of instances of an application queue, providing greater availability. Because you can define instances of the same queue on more than one queue manager, the workload can be distributed throughout the queue managers in a cluster. Queues that are used in this way need to be defined with `DEFBIND(NOTFIXED)` and opened with the open option `MQBND_BIND_NOT_FIXED`. The workload balancing algorithm can be used to allow you to 'fail over' to another queue manager. For example, this could be achieved by suspending one queue manager in a cluster hosting an instance of an application queue in the cluster and resuming another queue manager in the cluster hosting a second instance of the queue in the same cluster. In this situation, adopt a strategy to allow the System Cluster Transmit Queue (SCTQ) of the sending queue manager to drain before suspending the original queue manager from the cluster and resuming the second. This switch over to an alternative queue manager can form part of a planned outage for a queue manager for example. Additional information about clustering and workload balancing can be found in the WebSphere MQ 7.1 Information Center.

### **21.2.5 How can a message be processed by a different queue manager or queue without changing an application?**

Many WebSphere MQ applications might have originally been written with the queue or queue manager name hardcoded. It can be expensive and not always possible to rewrite these applications so that the target queue or queue manager can be changed. WebSphere MQ allows you to switch the queue or queue manager name in such situations with relative ease. This can also be useful when a queue manager needs to be stopped for a planned outage, but continued queue availability is required for the applications.

#### **Alias queue and queue manager definitions**

In an application, the use of alias queues or queue managers enable system administrators to easily change the definition of an alias object without having to change the application. For example, an application specifies the name of an alias queue or queue manager, defined on the queue manager to which it is connected, when it makes an `MQOPEN` request. When the application puts out a message, it is routed to the actual target queue or queue manager.

If the alias definition is for a queue, then the hosting queue manager resolves the real queue name, which can be a remote queue definition. Alternatively, if the alias is a queue manager alias, the message is routed to a queue on the target queue manager. An alias definition that is defined this way can be altered to reroute the messages to an alternative queue or queue manager without needing to alter the application. This is useful not only for workload balancing, but also to allow for planned maintenance outages while continuing to maintain application availability.

## 21.2.6 Improving availability to client applications during a planned outages

A *WebSphere MQ client* is a component of the WebSphere MQ product that can be installed on a system on which no queue manager runs. It enables an application, running on the same system as the WebSphere MQ client, to connect to a queue manager that is running on another system and issue Message Queue Interface (MQI) calls to that queue manager. Such an application is called a *WebSphere MQ client application* and the queue manager is referred to as a *server queue manager*. More information about clients versus servers can be found in the IBM WebSphere MQ Information Center.

In a WebSphere MQ client/server environment, there is unlikely to be an obvious distinction between planned and unplanned outages. A number of features can be used to increase availability through faster detection of connection failures and orphaned server-connection channels, and provide the ability for a client to try a connection to a different queue manager. This section describes these features:

- ▶ Client connection groups
- ▶ Duplexing

### Client connection groups

A queue manager group is a set of connections defined in the client channel definition table (CCDT). The set is defined by its members having the same value of the QMNAME attribute in their channel definitions. You can provide a number of destination queue managers to choose from to provide redundancy and alternate destinations when one fails or is no longer available as part of a planned outage.

### Duplexing

The SHARECNV parameter, specified when defining a TCP/IP channel, signifies the number of sharing conversations made available for channel and message processing. The value resolves to the lowest set between a CLNTCONN and SVRCONN pair. Provided SHARECNV does not resolve to zero on the channel, clients use full-duplex protocols for TCP/IP, enabling heart beat in both directions so that either end can detect loss of connection quickly. A client can then issue a new MQCONN or MQCONN and another connection in the connection list defined in the CCDT is tried. For client connections where sharing conversations are not in use, heartbeats flow from the server only when the client issues an MQGET call with wait.

**Note:** To take full advantage of alternate client connectivity, where possible, remove any message affinities. These are message transactions that rely on connection to a specific queue manager. Removing message affinities improves the availability and scalability of applications in general.

## 21.2.7 Further information

For a good starting point regarding queue manager in a QSG, see SupportPac MD17, WebSphere MQ for z/OS Highly Available System Design Version 1.0. However, the latest features are documented in the WebSphere MQ 7.1 Information Center at:

<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/index.jsp>

These are some helpful IBM Redbooks/Redpapers publications:

- ▶ *WebSphere MQ Primer: An Introduction to Messaging and WebSphere MQ*, REDP-0021
- ▶ *WebSphere MQ V6 Fundamentals*, SG24-7128
- ▶ *MQSeries Backup and Recovery*, SG24-5222

Also, see New York/New Jersey IBM MQ & Application Integration User Group presentations at [www.nynjmq.org](http://www.nynjmq.org).

## 21.3 Applying maintenance during subsystem operation

The following are key considerations when applying maintenance during subsystem operation:

- ▶ Concurrent code levels
- ▶ Do you need to bind DB2 application plans following maintenance?
- ▶ Catering to a DB2 outage in a QSG
- ▶ Performing maintenance on a queue manager in a cluster
- ▶ How can a queue manager be made to slow the rate of messages in the event of an IMS message flood?

### 21.3.1 Concurrent code levels

Queue managers on the same system can potentially run at different levels of WebSphere MQ code concurrently. However, exactly which levels of code that can run together, in a QSG for example, depends on a number of migration and toleration PTFs being in place. The details of these can be found in the latest documentation. A good place to start is “z/OS: Planning for migration” topics in the WebSphere MQ 7.1 Information Center. In addition, in a QSG, the queue manager code should have in place PTFs to allow WebSphere MQ for z/OS to support DB2 package versions. See the following section.

This means that it is possible to apply maintenance to a queue manager code base, but schedule and perhaps stagger the necessary queue manager restarts to pick up the latest code.

### 21.3.2 Do you need to bind DB2 application plans following maintenance?

When configuring WebSphere MQ for z/OS in a QSG, you must provide an environment in which the queue manager, channel initiator, and utilities can access and run against the required DB2 plans. As part of that procedure, you are required to customize and run a sample JCL to bind the DB2 plans.

In the past, if a PTF contained changes to code modules that included executable SQL statements, it was necessary to issue an extra BIND PLAN on one or more DB2 application plans used by WebSphere MQ for z/OS. Such a PTF includes a hold action of DB2BIND. In this case, only plans that were specified in the DB2BIND hold data must be rebound. The bind introduces a new plan to replace a corresponding old plan. This meant that queue managers that were up and running can continue to use the old plan and need not be restarted until suitably scheduled.

This approach also avoided a failure of the bind due to resource contention that might be encountered when rebinding an application plan (where a suspend operation or restart of the queue manager might be necessary).

However, the problem with this approach is that it was unclear and meant that new JCL must be customized each time this maintenance was shipped. It also had the disadvantage of needing to introduce a new plan.

## Avoidance of resource contention with DB2 Package Versioning

Following PTFs UK78376 (APAR PM60589) for WebSphere MQ for z/OS Version 7.1.0 and UK82506 (APAR PM65422) for WebSphere MQ for z/OS Version 7.0.1, each new DBRM included in a PTF introduces a new package version at BIND PACKAGE time.

Following the application of this maintenance, there are no new plans introduced with maintenance upgrades and only a BIND PACKAGE needs to be run for each DBRM shipped. Queue manager restarts, to pick up such maintenance updates, can be scheduled and staggered. Until a queue manager is restarted, it continues to run with the old code and against the old package version. Queue managers that are restarted to pick up the new code and package version can run concurrently with queue managers still running against the old code and package version within the same QSG.

**Important:** Although the queue manager does not need to be restarted following a BIND PACKAGE, for a queue manager to pick up a code change it must eventually be restarted. However, this provides flexibility when planning queue manager maintenance upgrades. It also allows concurrent versions of queue manager code to be run when using the same DB2 installation. In addition, the introduction of package versioning helps to avoid the need to suspend the queue manager regarding DB2 (see “Background” on page 270). In the past, this was sometimes necessary to avoid resource contention for the plan.

## Background

A PTF that includes a database request module (DBRM) needs to be bound to DB2 in such a way as to make its SQL and other host information available to an application plan for access at run time. Before WebSphere MQ for z/OS V 7, it was necessary to bind such DBRM directly to the plan with the BIND PLAN statement, creating a DBRM-bound plan.

A DBRM is a data set that contains SQL statements and host variable information that is extracted from the source program during program preparation. The purpose of a DBRM is to communicate SQL requests within executable code to DB2 during the bind process. A BIND uses the DBRM as input and produces either a package (containing a set of SQL statements from the DBRM) or an application plan. However, DBRM bound packages are deprecated in DB2 Version 9.1. As of WebSphere MQ for z/OS V 7, only DB2 package collections are bound to plans for queue manager execution. Packages, like DBRM, contain executable SQL. Unlike DBRM, they are held in indexed DB2 system catalog tables.

Packages can be logically grouped into collections. Which collection a package belongs to is determined when the package is bound with the BIND PACKAGE statement and signified by the collection name in the SYSIBM.SYSPACKAGE table. WebSphere MQ for z/OS associates its packages with a particular application plan by binding its corresponding collection to the plan. Each WebSphere MQ for z/OS plan only needs to be bound once in this way. Following the BIND PLAN, further updates to the plan from replacement DBRM only require a BIND PACKAGE to be issued to update the package in the collection.

Although this prevents the need to rebind the application plan following maintenance, such a BIND PACKAGE is as likely to cause contention for the plan, as it is binding the plan itself. This is because DB2 might hold locks for the package resource if queue managers are currently using the plan in question. To prevent this, new plans continued to be introduced in WebSphere MQ for z/OS V 7.0.1 until Package Versioning was introduced into the PTF build process.

Following PTFs UK78376 (APAR PM60589) for WebSphere MQ for z/OS Version 7.1.0 and UK82506 (APAR PM65422) for WebSphere MQ for z/OS Version 7.0.1, each new DBRM included in the PTF introduces a new package version at BIND PACKAGE time.



Following the application of this maintenance, no new plans are introduced and only a BIND PACKAGE needs to be issued for each DBRM.

This is because multiple versions of packages are allowed to coexist in a collection within an application plans. Therefore, several versions of the same module to which the DBRM corresponds can be run against the same plan concurrently. In addition, after a collection is bound to a plan, future packages or package versions that are bound to the collection are available for processing immediately. The plan does not need to be rebound when a package or package version is added to a collection. New packages or package versions can be added to a plan even while a plan is in use because the bind of the new package version does not interfere with an existing DB2 resource lock held for the plan.

For an in-depth description of DB2 packages, see *DB2 9 for z/OS: Packages Revisited*, SG24-7688. Further documentation can be found in the DB2 for z/OS Library:

<http://www.ibm.com/software/data/db2/zos/library.html>

### 21.3.3 Catering to a DB2 outage in a QSG

In a situation where you need to plan a DB2 subsystem outage, you can use the **SUSPEND QMGR FACILITY(DB2)** queue manager command to stop the queue manager connection to DB2, but allow the queue manager to continue to process messages that do not require access to DB2.

This command can also be used to prevent contention when performing a DB2 BIND PLAN of a WebSphere application plan that might be necessary following the application of maintenance. However, it is no longer necessary to suspend the queue manager for DB2 following the introduction of DB2 package versions for use with WebSphere MQ application plans.

When the queue manager is suspended with **SUSPEND QMGR FACILITY(DB2)**, there is no access to DB2 resources until the facility is resumed for the queue manager with **RESUME QMGR FACILITY(DB2)**.

Subsequent MQPUTs and MQGETs requiring access to the DB2 are suspended until the queue manager is resumed and connects to DB2 again. For example, large messages that are offloaded to DB2. WebSphere MQ for z/OS 7.1 introduced the use of Shared Message Data Sets (SMDS) as an alternative to DB2 for storing large messages on a shared queue. Where SMDS is configured for all large message offloads, such messages can still be processed when the queue manager is suspended for DB2.

**Note:** A number of operations will not work when the queue manager is suspended for DB2. These include displaying, deleting, altering, or defining a shared queue or group object definition. Issuing MQSET on a shared queue or group object definition prevents triggering and performance events from working correctly. Although shared channels continue to processing messages, a change such as starting or stopping will not work. For these reasons, carefully plan the use of SUSPEND QMGR FACILITY(DB2).

If you specified a DB2 group attach name in the QSGDATA parameter of the CSQ6SYSP system parameter module and there is another available member of the same data sharing group on the same z/OS image, you might be able to resume the queue manager and reconnect to the alternate DB2 subsystem, as in this example:

```
In MQ: SUSPEND QMGR FACILITY(DB2)
In one DB2: -STOP DB
In MQ: RESUME QMGR FACILITY(DB2)
```

### 21.3.4 Performing maintenance on a queue manager in a cluster

You can use a **SUSPEND QMGR** and **RESUME QMGR** command to temporarily reduce the inbound cluster activity to the queue manager with the **CLUSTER** or **CLUSNL** parameters.

When a queue manager is suspended from a cluster, it will not receive messages on instances of cluster queues that it hosts if there is another instance of the queue on an alternative queue manager in the cluster. This means that in an environment where message affinities are removed, and every application queue in the clustering environment has an instance on an alternative queue manager, queue managers can be stopped for maintenance with little or no impact on application availability.

When there are message affinities, such as the suspended queue manager is the target of the message or there is no other instance of the queue on an available queue manager in the cluster, messages will still be directed to the suspended queue manager.

Receiving further inbound messages while the queue manager is suspended can be prevented by stopping the cluster receiver channels for this cluster. To stop the cluster receiver channels for a cluster, use the **FORCE** mode of the **SUSPEND QMGR** command.

### 21.3.5 How can a queue manager be made to slow the rate of messages in the event of an IMS message flood?

In the past, an IMS system might have experienced a failure due to a shortage in system resources when too many client messages were queued. WebSphere MQ for z/OS offers a dynamic means of handling this problem rather than having to stop the queue manager.

#### IMS message floods

IMS might have to deal with many transaction requests within a short period. In some situations, not enough system resources exist to process all the requests at once. This situation is called a message flood. IMS OTMA is able to detect message thresholds and issue flood warning messages to clients in response.

The **SUSPEND/RESUME QMGR FACILITY(IMSBRIDGE)** support allows the IMS Bridge to be suspended for new requests to OTMA while allowing replies to come back. This works for both shared and non-shared queues, and provides a mechanism for manually throttling messages to the bridge.

As of WebSphere MQ for z/OS 7.1, a queue manager can also automatically respond to IMS OTMA flood warnings by throttling the rate at which messages are passed to IMS. This process can suspend all TPIPEs in the event of a full flood until the condition has been relieved. This process can help alleviate demands on ECSA storage by queuing such requests in the queue manager instead. For shared IMS bridge queues, this work can potentially be processed by a different queue manager and IMS.

## 21.4 Changes that can be made dynamically

This section describes changes that can be made while the queue manager is running and processing messages.

### 21.4.1 Can the buffer pools be changed while the queue manager is up?

For performance reasons, a queue manager stores messages for non-shared queues (and objects) in buffers (in private virtual storage) before writing them to page sets. Some shorter-lived messages might never be written to page sets. The buffers are organized into buffer pools.

If buffer pools are not tuned to the work load expected of the queue manager when processing messages on private queues, performance can be seriously affected. Therefore, it is some times necessary to make changes while a queue manager is still active, such as if there is an increase in messages being sent to or received from the queue manager due to a sudden increase in business demand.

The number of buffers in a buffer pool can be dynamically changed by an administrator. That is, the size of a buffer pool can be changed while the queue manager is processing messages with the use of the **ALTER BUFFPOOL** command. In addition, more buffer pools can be added with the **DEFINE PSID** command by using the **DSN** keyword to specify a linear VSAM data set.

You need to check how much space is available for extra buffer pool usage by checking the latest CSQY220I messages in the queue manager log. The available space is reported in MB or GB of free local storage. Because each buffer is 4 KB, each MB of free space allows you to allocate a further 256 buffers. However, leave some free space for other processing requirements of the queue manager.

**Note:** If you do change the number of buffers in the buffer pool, also consider changing the **DEFINE BUFFPOOL** commands in the CSQINP1 initialization input data set used by the queue manager. This is needed so that the changes remain in force when the queue manager is restarted.

## 21.5 Changes requiring a queue manager restart or IPL

This section details the changes that require a queue manager restart or a system IPL to be made on the system.

### 21.5.1 Does an early code upgrade need a queue manager restart or IPL?

To create a queue manager subsystem, you must have code that is loaded into the z/OS link pack area (LPA). The modules that contain this code are often referred to as the early code. This code is executable even when the queue manager is not running. It starts the queue manager when an operator enters the start command, for example.

There can only be one active set of early code modules within a Sysplex. Every queue manager runs from this same set of early code modules. To run different queue manager versions on the same Sysplex, the latest level of early code is required. This is because early code is backward compatible, but old levels of early code are not supported for later levels of queue manager code.

After upgrading the early code, it can be made available immediately without an IPL by adding it to the LPA with the **SETPROG LPA,ADD** command. This process means that a new queue manager subsystem can be dynamically defined with the system command **SETSSI ADD**. Existing queue manager subsystems can pick up the new code with the **REFRESH QMGR TYPE(EARLY)** command, provided the pre-existing early code was Version 5.3 or later.

You must also ensure that all subsystems are defined in the IEFSSNxx members in SYS1.PARMLIB, so they are also available after an IPL.

**Note:** The Early code is designed in such a way as to minimize the need for an IPL following maintenance upgrades. However, a queue manager must be stopped to issue **REFRESH QMGR TYPE(EARLY)** against it. That means that maintenance that includes Early code requires the queue manager to be restarted to pick up the code change.

Sometimes, a code change means that the code is only run when a queue manager subsystem is defined. Queue manager subsystems cannot be dynamically deleted or modified. So, for an existing queue manager subsystem, an IPL is needed to pick up the changes for this special case of Early code. A queue manager subsystem that is added dynamically with the **SETSSI** command does not require an IPL in this situation.

## 21.6 Error recovery (without a system or subsystem restart)

The queue manager can recover from various errors it detects, including program checks, full data sets, and internal consistency errors. This section describes a number of features and facilities that help in this situation and allow for minimal impact to application availability.

### 21.6.1 What happens to coordinated unit of recovery in a QSG?

For queue managers in a QSG, WebSphere MQ for z/OS Version 7.0.1 includes extra transactional facilities for XA coordinated transactions. Queue managers in a QSG can be configured to run GROUP units of recovery. When enabled for each queue manager in the QSG, any queue manager in the QSG can run recovery of an in-doubt transaction on behalf of an XA client.

An XA client application connecting to a QSG might connect without specifying a particular queue manager. This configuration means that if the client needs to reconnect to the QSG and resolve an in-doubt unit of recovery and the queue manager to which the application was initially connected is not available, the recovery can still take place.

This feature removes the dependency of an XA client application having to reconnect to a specific member of the QSG and extends the availability of shared application queues. It also helps reduce the impact of a planned outage for a queue manager in a QSG.

WebSphere MQ for z/OS V7.1 extends these availability benefits with GROUP units of recovery for CICS applications. CICS Transaction Server V4.2 introduced the group resynchronization option, **RESYNCMEMBER(GROUPRESYNC)**, to the **MQCONN** definition. In such a situation, CICS can connect to any queue manager in a QSG on the same image and its transactions run as GROUP units of recovery. In-doubt units of recovery between CICS and the QSG can be resolved when CICS reconnects to any available queue manager in the QSG on the same image. This enhancement removes an affinity between CICS regions and queue managers, which simplifies system configuration and enhances availability.

### 21.6.2 WebSphere MQ peer recovery in a QSG

Queue managers working in a QSG, in many cases, are able to take over work from a failed peer.

### **Application structure peer recovery**

WebSphere MQ can detect if a queue manager in the QSG abnormally disconnects from the CF. In such situations, where possible, another queue manager in the QSG runs peer recovery and completes any outstanding units of work on behalf of the failed queue manager.

Units of work that were in doubt when the failure occurred will not be recovered. There are strategies using the z/OS Automatic Restart Manager that can be adopted to help automate recovery in these situations, further reducing downtime for applications due to unplanned outages.

### **Admin structure peer recovery**

The administration structure is used to hold status information that is used by the queue managers in the QSG to coordinate work and manage recovery. Actual message data is held in application structures. Each queue manager logs any recoverable update that it makes to the administration structure in its own log. From WebSphere MQ Version 7.0.1, any queue manager is able to rebuild the entries for any other queue manager in the QSG. Before this version, if the administration structure needed recovering, a queue manager automatically rebuilt its own entries from its log, and needed to be active to do so.

For more information about Peer recovery within a QSG, see the WebSphere MQ Version 7.1 Information Center:

<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/index.jsp>

## **21.6.3 Toleration for CF connectivity loss and recovery enhancements**

With the addition of CF structure attributes CFCONLOS and RECAUTO at WebSphere MQ for z/OS V7.1, in the event of a CF outage, each queue manager is able to continue processing private queues. Shared queue application downtime is limited to the time required to rebuild or automatically recover CF structures.

### **Toleration for Coupling Facility connectivity loss**

From WebSphere MQ for z/OS V7.1, toleration of CF connection failure is available with the introduction of the CFCONLOS structure attribute. This attribute uses z/OS system managed rebuild facilities to automatically rebuild a CF structure after a structure failure or a total loss of connectivity for a structure that it is connected to. This parameter is only valid from CFLEVEL 5 or later. When set to Tolerate, loss of connectivity to the CF structures will not stop the queue manager.

#### ***The admin structure***

During loss of connectivity to the administration structure, when CFCONLOS is set to Tolerate, all the active queue managers in the QSG disconnect from the admin structure. Each queue manager then attempts to reconnect and rebuild its admin structure data. If a queue manager cannot reconnect to the admin structure, for example because there is no CF with connectivity available, the queue manager runs with reduced shared queue functionality. The queue manager continues to process private queues. The queue manager will reconnect after the CF is available again.

#### ***Application structures***

Each queue manager that loses connectivity to an application structure disconnects from the structure. If there is still a system in the sysplex that has connectivity to the CF that the structure is allocated in, the queue manager that lost connectivity to the structure attempts to initiate a system-managed rebuild. The rebuild moves the structure to another CF with better connectivity. If an application structure cannot be reallocated in another CF with better

connectivity, queues on the structure remain unavailable until connectivity is restored to the CF that the structure is allocated in. During a total loss of connectivity, the structure might need to be recovered manually with the **RECOVER CFSTRUCT** command. The queue manager automatically reconnects to the structure when it becomes available.

### **Automatic structure recovery**

With the RECAUTO CF attribute set to YES, introduced in WebSphere MQ for z/OS V7.1, structures can also be recovered if detected to be failed or empty at the time of connection. This feature also ensures that a queue manager attempts to connect to all the structures in the QSG during start. This means that any necessary recoveries are run when a queue manager starts or when any active queue manager detects a failed or empty structure.



# A

## Items requiring an IPL or recycle

This appendix contains a list of items that cannot be modified dynamically and require an IPL or a recycle to implement. Because the *z/OS Planned Outage Avoidance Checklist*, SG24-7328 was written in 2006, many things have changed. These allow for dynamic changes in both the hardware and software arenas, as well as different processes to be implemented to improve availability in the subsystem world.

This appendix contains this section:

- Items requiring an IPL or recycle to implement

## Items requiring an IPL or recycle to implement

Despite the improvements that have been made to the operating system, its components, and its subsystems over the last few years, some changes still require an IPL or recycle. Table A-1 contains a list of the most common such changes. There is also information about where to find additional information about parameters and whether they can be activated dynamically in your particular release and for your particular system or subsystem.

One of the objectives in a multi-system environment is to share PARMLIB members with as many LPARs as possible within a sysplex. You might be able to use system symbolics to accomplish this.

*Table A-1 List of items still requiring an IPL*

| Component        | Item                                                                                                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|-----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Operating System | JES3: add spool volume                                                                                          | Requires a warm start of JES3 and a sysplex IPL to add a spool volume. This was changed in 1.13 to allow dynamic adds of spool volumes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Operating System | RACF: new ICHRDSNT and ICHRRNG                                                                                  | Changes to the RACF data set name table (ICHRDSNT) and RACF range table require a sysplex IPL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Operating System | CONSOLxx in SYS1.PARMLIB                                                                                        | You cannot add a hardware console without an IPL. This process was made dynamic in V2.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Operating System | Backout of GRS Star                                                                                             | Although you can move from GRS Ring to GRS Star dynamically, a sysplex IPL is needed to go back to Ring mode, if necessary.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Operating System | MAXUSER, RSVSTRT, and RSVNONR parameters                                                                        | These IEASYSxx parameters can only be modified by using an IPL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Operating System | IBM service such as HOLD(IPL) and HOLD(Restart). SMP/E now gives you this information for all maintenance runs. | To apply maintenance to modules that are in LPALIB usually requires an IPL. When it is necessary, the HOLDDATA instructions (HOLDDATA) that accompany the PTFs document this. These document cover maintenance of subsystems such as the USS read-only root file system, JES2, and TCP/IP.<br>Although it is technically possible to apply maintenance to these modules dynamically, many sites find it easier to close the system down and restart because of the complications involved with closing all applications using any of these subsystems.<br>Although many sites have automation in place to close a system down cleanly and then restart it, far fewer have similar automation to stop and start individual subsystems. |
| UNIX             | USS SWA() parameter                                                                                             | Changing the SWA() parameter in BPXPRMxx requires an IPL to implement.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| UNIX             | USS SYSPLEX() parameter                                                                                         | Changing the SYSPLEX() parameter in BPXPRMxx requires an IPL to implement.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| UNIX             | MODIFY OMVS,SHUTDOWN                                                                                            | There are instances where <b>F OMVS,SHUTDOWN</b> cannot shut down z/OS UNIX completely and an IPL is still required to correct the condition that instigated the shutdown.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Operating System | LPA modules                                                                                                     | It is usually necessary to run an IPL to replace LPA modules. For example, many service updates of LPA modules require an IPL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |




| Component        | Item                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Operating System | BRLM implementation | Moving to the Distributed Byte Range Lock Manager (BRLM) can be done dynamically, but a sysplex-wide IPL is required to back out this change.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Operating System | PDSE sharing        | Reverting to PDSESHARING(NORMAL) requires a sysplex IPL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Operating System | New master catalog  | Changes to either the LOADxx member of PARMLIB or the SYSCATLG member in SYS1.NUCLEUS require an IPL to pick up the change.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Operating System | System symbols      | An IPL is required to implement or modify system symbols. IEASYMUP (up through z/OS 1.13) can be used for this purpose, but it is not fully supported and should be used with caution. V2.1 introduces the <b>SETLOAD xx, IEASYM</b> command that can dynamically update System Symbols. This command can be found in the z/OS section on Symbols in this book.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Operating System | PARMLIB members     | <p>The following PARMLIB members can only be updated with an IPL to pick up changes:</p> <ul style="list-style-type: none"> <li>▶ ALLOCxx (This member was made into a dynamic command in 1.11. After an IPL, use the SETALLOC command to change any of the defaults except for 2DGT_EXPDT.)</li> <li>▶ BLSCECT</li> <li>▶ BLSCUSER</li> <li>▶ CNIDTRxx (This member is gone in V2.1, and was replaced by GTZPRM, which is dynamic)</li> <li>▶ CONFIGxx (The CONFIG command with the member option enables an operator to reconfigure the system according to the options in the specified CONFIGxx member, CF member=xx). This command can modify the following settings: Change online, offline status of available processors, storage sections, and channel paths.</li> <li>▶ DEVSUP (After an IPL, you can use the system command <b>SET DEVSUP=xx</b> to activate the DEVSUP changes. This was made dynamic in V1.8.)</li> <li>▶ EPHWP00</li> <li>▶ IEAAP00</li> <li>▶ IEAFIX00</li> <li>▶ IEAPAKxx</li> </ul> |

| Component        | Item                                                                                                                      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Operating System | PARMLIB members (continued)                                                                                               | <ul style="list-style-type: none"> <li>► IOEPRMxx<br/>Contains the configuration options that control the z/OS Distributed File Service z/OS file system (zFS) environment and zFS aggregates to be attached at the start of zFS. The IOEPRMxx files are contained in the logical PARMLIBconcatenation, which is defined on the PARMLIB parameter in LOADxx member. Alternatively, you can specify the zFS configuration options in the IOEFSPRM file, which is pointed to by the IOEZPRM DD statement in the ZFS PROC. For information about parameters for the IOEPRMxx PARMLIB member and the IOEFSPRM file, see IOEFSPRM in <i>zOS Distributed File Service zSeries File System Administration</i>, SC24-5989.</li> </ul> <p>The command <b>Modify ZFS</b> can change many options/statements. More detail on this command can be found in <i>zOS Distributed File Service zSeries File System Administration</i>, SC24-5989.</p> |
| Operating System | PARMLIB members continued                                                                                                 | <ul style="list-style-type: none"> <li>► IEASYSxx</li> <li>► LOADxx<br/>For all parameters other than PARMLIB concatenation. DYNCPADD indicates whether z/OS support for dynamic CPU addition is enabled. This option was introduced with v1.10. For more information, see <i>z/OS MVS Initialization and Tuning Reference</i>, SA23-1380, for any release after v1.9</li> <li>► IMSTJCLxx</li> <li>► NUCLSTxx</li> <li>► SCHEDxx (For all parameters other than PPT entries)</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Operating System | PLPA and COMMON PAGE data sets                                                                                            | Amendments to these page data sets require an IPL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Operating System | SYSPLEX enablement                                                                                                        | Moving from XCFLOCAL mode to MONOPLEX or MULTISYS mode requires an IPL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Operating System | New operating system (for example upgrading from z/OS 1.12 to z/OS 1.13)                                                  | Because all the base components of an operating system are changed with a new release, an IPL on a new SYSRES is required                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Operating System | Infoprint Server                                                                                                          | Many attributes are dynamic as of z/OS V2R1. However, some attributes require one or more daemons to be stopped and restarted. The attribute aophinvd-max-thread-tasks requires that all daemons are stopped and restarted. For a complete list of all dynamic attributes that require no restart or some daemon restarts, see <i>z/OS Infoprint Server Customization</i> , SA38-0691.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Operating System | AXRxx—default System REXX data set default concatenation. <i>z/OS MVS Initialization and Tuning Reference</i> , SA23-1380 | AXRxx is a member of SYS1.SAMPLIB. It can be copied to SYS1.xx.PARMLIB when installation changes to default settings are required. For more information about this option, see <i>z/OS MVS Initialization and Tuning Reference</i> , SA23-1380, for your particular release.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| CICS             | New release                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CICS             | Maintenance to current release                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

| Component | Item                           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CICS      | Non-dynamic system parameters  | A number of system parameters can be updated dynamically. See Chapter 18, “CICS considerations” on page 183 and the <i>CICS Transaction Server for z/OS Version 5 Release 1 Resource Definition Guide</i> , SC34-2848, for your particular release to discover which dynamic parms are available. For the CICS Transaction Server, the syntax of the system init parms that can be coded in the DHFHSIT macro. For more information, see the <i>CICS Transaction Server for z/OS Version 5 Release 1 Resource Definition Guide</i> , SC34-2848, for your particular release. |
| DB2       | New release                    | 19.1.2, “Version to version upgrade/fallback” on page 226                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| DB2       | Maintenance to current release | 19.1.1, “Applying DB2 service (PTF/RSU/CST)” on page 216                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| DB2       | Non-dynamic system parameters  | <p>Most DSNZPARMs can be updated dynamically, but a small number do require that DB2 be recycled. See 19.1.3, “DSNZPARM changes” on page 227.</p> <p>In addition, the DB2 xxx manuals need the variable for your particular release to discover which online changeable DSNZPARM values are available. They are found under a topic similar to “Directory of subsystem parameters and application default values” under a column called “Update online”.</p>                                                                                                                 |
| WAS       | New release                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| WAS       | Maintenance to current release |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| WAS       | Non-dynamic system parameters  | <p>Configuring the application server requires three steps:</p> <ol style="list-style-type: none"> <li>1. Set the app server type and common properties.</li> <li>2. Configure the parameters for the appserver type.</li> <li>3. Configure the parameters for each appserver service.</li> </ol> <p>More information about these parms can be found in the Install manuals for your particular release of WebSphere. Some parms can be set dynamically and are described in detail in the Install manuals.</p>                                                              |
| IMS       | New release                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| IMS       | Maintenance to current release |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| IMS       | Non-dynamic system parameters  | A number of System parameters that can be issued or updated dynamically for IMS. See Chapter 20, “IMS considerations” on page 251 in this book and the appropriate <i>IMS Version 13 Installation</i> , GC19-3656, or the appropriate manual for your particular version for more details                                                                                                                                                                                                                                                                                    |
| MQ        | New release                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| MQ        | Maintenance to current release |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| MQ        | Non-dynamic system parameters  | When you use MQ messaging over a network, you must set several database server configuration parameters. Some of these might be dynamic. See Chapter 21, “WebSphere MQ for z/OS” on page 263 in this book and the appropriate WebSphere MQ Manual for your release for more details.                                                                                                                                                                                                                                                                                         |





## **DB2 UNIX file system maintenance considerations**

This appendix shows an example of flexible DB2 maintenance from the perspective of the file system. The file system contains the IBM Data Server Driver for JDBC and SQLJ.

This appendix contains this section:

- ▶ “Type 2 and type 4 connectivity” on page 284
- ▶ “Example of DB2 maintenance and the zFS” on page 284

## Type 2 and type 4 connectivity

The IBM Data Server Driver for JDBC and SQLJ supports two types of connectivity: Type 2 and type 4. The following points can help determine which connectivity to use.

Use IBM Data Server Driver for JDBC and SQLJ type 2 connectivity under these circumstances:

- ▶ Your JDBC or SQLJ application runs locally most of the time.  
Local applications have better performance with type 2 connectivity.
- ▶ You are running a Java stored procedure.  
A stored procedure environment consists of two parts: A client program, from which you call a stored procedure, and a server program, which is the stored procedure. You can call a stored procedure in a JDBC or SQLJ program that uses type 2 or type 4 connectivity, but you must run a Java stored procedure using type 2 connectivity.
- ▶ Your application runs in the CICS environment or IMS environment.
- ▶ Your application runs in WebSphere for z/OS environment.

Use IBM Data Server Driver for JDBC and SQLJ type 4 connectivity under these circumstances:

- ▶ Your JDBC or SQLJ application runs remotely most of the time.  
Remote applications have better performance with type 4 connectivity.
- ▶ You are using IBM Data Server Driver for JDBC and SQLJ connection concentrator and Sysplex workload balancing support.
- ▶ You can use type 4 connectivity with WebSphere for z/OS, but type 2 provides better performance.

To summarize:

- ▶ Type 2 is preferred for local access to DB2.
- ▶ Type 4 must be used for distributed access to DB2 for z/OS.
- ▶ Type 4 can be used for local access to DB2.

For type 2 connectivity, the Java portion of the driver is loaded from the zFS, whereas the native DLL portion of the driver loads from the SDSNLOAD2 data set. The SDSNLOAD2 service level must exactly match that of DB2.ZFS. Normally the service level of SDSNLOAD does not matter. For the STEPLIB concatenation, it does not matter whether SDSNLOAD or SDSNLOAD2 are concatenated behind SDSNEXIT, as the modules are different.

The example in this appendix works correctly for either type of connectivity.

## Example of DB2 maintenance and the zFS

This example illustrates the maintenance process that was introduced in “File system activities” on page 223. It was developed on an IBM Washington Systems Center Parallel Sysplex called WSCPLEX and is based on a 4-way DB2 data sharing group called WSCDBP0, with members DBP1-DBP4.

It demonstrates that DB2 maintenance levels can be switched transparently to the users, in a similar manner to the way the DB2 z/OS data set are switched.

The WSCPLEX has a shared file system.

- ▶ The root directory is /shared.
- ▶ The system-specific file systems are: /SYSA, /SYSB, /SYSC, /SYSD.
- ▶ There are four systems, but for the sake of simplicity, this example only covers two: SYSA, where DBP1 runs, and SYSB, where DBP2 runs.

There are two multi-system data sharing groups (WSCDBP0 and DB2QUAL) that can run on different DB2 service levels and even different DB2 versions on the two systems. The company wants the users to only know which data sharing group they are accessing from either SYSA or SYSB. They want them always to export the same PATH, LIBPATH, or CLASSPATH string regardless of which system they are on. For example, they want the users of WSCDBP0 group to export PATH=/shared/db2dbp0:\$PATH regardless of whether they run on SYSA or SYSB. They also want to ensure that the correct libraries are set up so that when they run their allocation script, it just works.

Similarly, they want to have the users of the DB2QUAL data sharing group, which might not be at the same version or service level as WSCDBP0, to export PATH=/shared/db2qual:\$PATH and if they run their allocation script, it just works.

There are two allocation scripts:

- ▶ **red\_dbp0.sh**, shown in Example B-4 on page 288
- ▶ **red\_qual.sh**, shown in Example B-5 on page 289

“File system activities” on page 223 shows three levels of service. To implement the 3-level approach, you also need a **I3** (previous service level), which is “DB2HQ1.DBP0.RSU1401.DB2.ZFS.” The example deals just with the two levels.

- ▶ The instance of WSCDBP0 on SYSA, DBP1 should use V1 of the code base (RSU1404) and the instance of WSCDBP0 on SYSB should use V1 (RSU1404).
- ▶ The instance of DB2QUAL on SYSA, DBP2 should use V2 of the ZFS (RSU1407) and the instance of DB2QUAL on SYSB should use V2 (RSU1407) of the code.

Create a number of symlinks that will resolve to the correct level of code.

In the /shared file system, create two mount points: “jrb\_l1” and “jrb\_l2”.

- ▶ Mount “DB2HQ1.DBP0.RSU1404.DB2.ZFS” (the current service level) at /shared/jrb\_l1
- ▶ Mount “DB2HQ1.DBP0.DB2.RSU1407.ZFS” (the new service level) at /shared/jrb\_l2

Now, add some indirection. In the /SYSA file system, create two symlinks:

```
jrb_l1 --> /shared/jrb_l1
jrb_l2 --> /shared/jrb_l2
```

Then, do the same in the /SYSB file system.

In the /SYSA file system, create the symlink “which\_qual” and point it at “jrb\_l2”. Also, create the symlink “which\_dbp0” and point it at “jrb\_l1”.

The resulting file systems looks as shown in Example B-1:

#### *Example B-1 SYSA DB2 file system symlinks*

---

```
wsc1:/SYSA
-> ll
total 296
drwxrwxrwx  5 QWER01  GROUP0    8192 Aug  1 14:44 tmp
```

```

drwxr-xr-x   6 QWER01  SYS1          8192 Aug  1 10:01 .
lrwxrwxrwx   1 BROWN   SYS1           6 Aug  1 10:01 which_qual -> jrb_l2
lrwxrwxrwx   1 BROWN   SYS1          14 Aug  1 09:59 jrb_l2 -> /shared/jrb_l2
lrwxrwxrwx   1 BROWN   SYS1          14 Aug  1 09:58 jrb_l1 -> /shared/jrb_l1
lrwxrwxrwx   1 BROWN   SYS1           6 Jul 31 17:05 which_dbp0 -> jrb_l1
drwxrwxrwx  43 PFAUSER  SYS1          8192 Jul 31 15:40 ..
drwxr-xr-x   2 QWER01  SYS1       110592 Jun 30 23:47 dev
drwxr-xr-x  22 QWER01  OMVSGRP      8192 Jun 30 22:24 etc
drwxr-xr-x  22 QWER01  SYS1          8192 Jul 22 2012 var
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 bin -> /bin
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 lib -> /lib
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 opt -> /opt
lrwxrwxrwx   1 QWER01  SYS1           8 Mar  6 2000 samples -> /samples
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 usr -> /usr
wsc1:/SYSA
->

```

---

Next, in the /SYSB file system, create the symlink “which\_qual” and point it at “jrb\_l2” and create the symlink “which\_dbp0” and point it at “jrb\_l1” as shown in Example B-2

*Example B-2 SYSB DB2 file system symlinks*

---

```

wsc2:/SYSB
-> ll
total 136
drwxrwxrwt   9 QWER01  GROUP0     12288 Aug  1 13:55 tmp
drwxr-xr-x   6 QWER01  SYS1          8192 Aug  1 09:59 .
lrwxrwxrwx   1 BROWN   SYS1          14 Aug  1 09:59 jrb_l2 -> /shared/jrb_l2
lrwxrwxrwx   1 BROWN   SYS1          14 Aug  1 09:59 jrb_l1 -> /shared/jrb_l1
lrwxrwxrwx   1 BROWN   SYS1           6 Jul 31 17:05 which_qual -> jrb_l2
lrwxrwxrwx   1 BROWN   SYS1           6 Jul 31 16:58 which_dbp0 -> jrb_l1
drwxrwxrwx  43 PFAUSER  SYS1          8192 Jul 31 15:40 ..
drwxr-xr-x  23 QWER01  SYS1       24576 Jul  1 00:00 etc
drwxr-xr-t   2 QWER01  SYS1          8192 Oct 27 2011 dev
drwxrwxrwt  18 QWER01  SYS1          8192 Oct 26 2011 var
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 bin -> /bin
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 lib -> /lib
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 opt -> /opt
lrwxrwxrwx   1 QWER01  SYS1           8 Mar  6 2000 samples -> /samples
lrwxrwxrwx   1 QWER01  SYS1           4 Mar  6 2000 usr -> /usr
wsc2:/SYSB
->

```

---

Now make “which\_dbp0” and “which\_qual” resolve correctly, depending on the system you are using. To do that, use the z/OS system symbol, \$SYSNAME. It resolves to the system name and is set at IPL time. For more information about system symbols, see 4.10, “Dynamically change system symbols” on page 70.

In the /shared file system, create a symlink “db2qual” and point it at /\$SYSNAME/which\_qual. Create a symlink “db2dbp0” and point it at /\$SYSNAME/which\_dbp0.



It looks something like Example B-3.

*Example B-3 db2dbp0 and db2qual symlinks*

---

```
wsc1:/shared
-> ll
total 774
drwxr-xr-x  2 BROWN  SYS1      8192 Aug  1 09:56 jrb_l2
drwxr-xr-x  2 BROWN  SYS1      8192 Aug  1 09:56 jrb_l1
drwxr-xr-x 51 QWER01  SYS1      8192 Aug  1 09:56 .
lrwxrwxrwx  1 BROWN  SYS1          20 Jul 31 17:44 db2dbpo ->
/$SYSNAME/which_dbp0
lrwxrwxrwx  1 BROWN  SYS1          20 Jul 31 16:58 db2qual ->
/$SYSNAME/which_qual
drwxrwxrwx 43 PFAUSER SYS1      8192 Jul 31 15:40 ..
wsc1:/shared
->
```

---

## Allocation scripts

These scripts provide the correct PATHs and STEPLIB for the Java workload.

The idea is that the application can run on either SYSA or SYSB with the correct service level and the script needs no changes. The output displayed is the third qualifier, which you use for the service level. It ensures that the PATH, LIBPATH, CLASSPATH have the STEPLIB libraries set correctly.

In “Executions of the scripts” on page 291, run the scripts that demonstrate that DBP1/DBP2 are at the RSU1404 level. DB2QUAL is at RSU1407. Then, upgrade DBP1 to RSU1407, while leaving DBP2 at the older service level RSU1404.

**Note:** It is not necessary for this example, but use the same set of libraries for the Washington Systems Center sysplexes.

In both scripts, the client should know only these items:

- ▶ The symlink that is used by the application
- ▶ The high-level qualifier of the DB2 system libraries, such as DB2HLQ1
- ▶ The second-level qualifier, normally the group attach name. It becomes the second-level qualifier of the DB2 system libraries. In this case, it is DBP0

The trick to providing the proper STEPLIBs is to determine the service level of the libraries. The script in Example B-4 parses the third qualifier of a DB2HLQ1.DBP0.xx.DB2.ZFS and uses it to create the STEPLIB concatenation.

```
level=`df ${JCC_HOME}/classes | grep -e '(' | awk '{print $2}' | cut -d . -f 3`
echo $level
```

Therefore, a prerequisite is that the DB2.ZFS must be mounted before the script is run.

## Script for WSCDBP0

Example B-4 shows the “red\_dbp0.sh” script. Note the bolded lines where the symlink, the DB2 high-level qualifier, and the second-level qualifier are set.

*Example B-4 Script to display the service level for WSCDBP0*

---

```
-> cat red_dbp0.sh
#!/bin/sh
#
# Set the environment variables correctly(?) for using the
# DB2 jcc with java 1.4 on the wscx system
#
# Run this shell script with the . (dot) command to set the environment variables
# for this session.
#

#
# set the location of the sdk home, the jcc home, the DB2 OS data set
# highlevel qualifier, and the current directory
#

if [ -z "$JAVA5_HOME" ]; then
    JAVA5_HOME=/shared/java/J5.0
fi
DB2_HOME=/shared/db2dbp0
JCC_HOME=$DB2_HOME/jdbc
DB2_HLQ=DB2HLQ1
DB2_SUBSYS=DBP0

BASE_DIR=`pwd`

#
# Add the jcc related jar files to the CLASSPATH
#
JCC_CLASSPATH=

# Add jars from JCC_HOME/classes for jcc
JCC_CLASSPATH=$JCC_HOME/classes/db2jcc.jar
JCC_CLASSPATH=$JCC_HOME/classes/sqlj.zip:$JCC_CLASSPATH
JCC_CLASSPATH=$JCC_HOME/classes/db2jcc_javax.jar:$JCC_CLASSPATH
JCC_CLASSPATH=$JCC_HOME/classes/db2jcc_license_cisuz.jar:$JCC_CLASSPATH

# Add jars from ${PWD}/lib

#for i in "${PWD}/lib"/*.jar; do
#    JCC_CLASSPATH="$i:$JCC_CLASSPATH"
#done

# Add the current directory to pick up classes anchored here, and concatenate

export CLASSPATH="$BASE_DIR":$JCC_CLASSPATH:$CLASSPATH

#
# Add the jcc related directories to the PATH and LIBPATH
#
```

```

export PATH="${JCC_HOME}"/bin:$PATH
export LIBPATH="${JCC_HOME}"/lib:$LIBPATH

#
# Add the java directory to the PATH if needed
#

whence java | grep -q -e "J5.0" -e "J6.0"
rc=$?

if [ rc -eq 1 ]; then
    export PATH="${JAVA5_HOME}"/bin:$PATH
fi

#
# If SDSNLOD2, SDSNLOAD and SDSNEXIT are not in the linklist,
# fixup the following statements and uncomment the export statement.
#

level=`df ${JCC_HOME}/classes | grep -e '(' | awk '{print $2}' | cut -d . -f 3`
echo $level
SL=
SL=${DB2_HLQ}.${DB2_SUBSYS}.SDSNEXIT
SL=$SL:${DB2_HLQ}.${DB2_SUBSYS}.${level}.SDSNLOAD
SL=$SL:${DB2_HLQ}.${DB2_SUBSYS}.${level}.SDSNLOD2

export STEPLIB=$SL:$STEPLIB

```

---

## Script for DB2QUAL

Example B-5 shows the script for the DB2QUAL data sharing group. Note the bolded lines where symlink, the DB2 high-level qualifier, and the second-level qualifier are set. Because DB2QUAL uses the same libraries as WSCDBP0, the DB2\_SUBSYS is "DBP0" versus the group attach name of DB2QUAL.

*Example B-5 Script to display service level for DB2QUAL group*

---

```

-> cat red_qual.sh
#!/bin/sh
#
# Set the environment variables correctly(?) for using the
# DB2 jcc with java 1.4 on the wsc4 system
#
# Run this shell script with the . (dot) command to set the environment variables
# for this session.
#

#
# set the location of the sdk home, the jcc home, the DB2 OS data set
# highlevel qualifier, and the current directory
#

if [ -z "$JAVA5_HOME" ]; then
    JAVA5_HOME=/shared/java/J5.0
fi

```

```

DB2_HOME=/shared/db2qua1
JCC_HOME=$DB2_HOME/jdbc
DB2_HLQ=DB2HLQ1
DB2_SUBSYS=DBP0
# subsys can be group attach qualifier of library you want
# true for DB2QUAL

BASE_DIR=`pwd`

#
# Add the jcc related jar files to the CLASSPATH
#
JCC_CLASSPATH=

# Add jars from JCC_HOME/classes for jcc
JCC_CLASSPATH=$JCC_HOME/classes/db2jcc.jar
JCC_CLASSPATH=$JCC_HOME/classes/sqlj.zip:$JCC_CLASSPATH
JCC_CLASSPATH=$JCC_HOME/classes/db2jcc_javax.jar:$JCC_CLASSPATH
JCC_CLASSPATH=$JCC_HOME/classes/db2jcc_license_cisuz.jar:$JCC_CLASSPATH

# Add jars from ${PWD}/lib

#for i in "${PWD}/lib"/*.jar; do
#    JCC_CLASSPATH="$i":$JCC_CLASSPATH"
#done

# Add the current directory to pick up classes anchored here, and concatenate

export CLASSPATH="$BASE_DIR":$JCC_CLASSPATH:$CLASSPATH"

#
# Add the jcc related directories to the PATH and LIBPATH
#

export PATH="$JCC_HOME"/bin:$PATH
export LIBPATH="$JCC_HOME"/lib:$LIBPATH

#
# Add the java directory to the PATH if needed
#

whence java | grep -q -e "J5.0" -e "J6.0"
rc=$?

if [ rc -eq 1 ]; then
    export PATH="$JAVA5_HOME"/bin:$PATH
fi

#
# If SDSNLOD2, SDSNLOAD and SDSNEXIT are not in the linklist,
# fixup the following statements and uncomment the export statement.
#

level=`df  ${JCC_HOME}/classes | grep -e '(' | awk '{print $2}' | cut -d . -f 3`

```

```

echo $level
SL=
SL=${DB2_HLQ}.${DB2_SUBSYS}.SDSNEXIT
SL=$SL:${DB2_HLQ}.${DB2_SUBSYS}.${level}.SDSNLOAD
SL=$SL:${DB2_HLQ}.${DB2_SUBSYS}.${level}.SDSNLOD2

export STEPLIB=$SL:$STEPLIB

```

---

## Executions of the scripts

Example B-6 shows the /SYSA file system and important symlinks before you run the WSCDBP0 script.

### *Example B-6 Environment for wsc1 (SYSA)*

---

```

wsc1:/u/judyrb
-> cd /SYSA
wsc1:/SYSA
-> ls -al which*
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 17:05 which_dbp0 -> jrb_11
lrwxrwxrwx  1 QWERO1 SYS1          6 Aug 12 07:24 which_qual -> jrb_12
wsc1:/SYSA
-> cd /shared/jrb_11
wsc1:/shared/jrb_11
-> df -v .
Mounted on      Filesystem              Avail/Total      Files      Status
/shared/jrb_11 (DB2HLQ1.DBP0.RSU1404.DB2.ZFS)  107962/144000   4294967214
Available
ZFS, Read/Write, Device:2896, ACLS=Y
File System Owner : SYSD      Automove=Y      Client=N
Filetag : T=off  codeset=0
Aggregate Name : DB2HLQ1.DBP0.RSU1404.DB2.ZFS
wsc1:/shared/jrb_11
-> cd /shared/jrb_12
wsc1:/shared/jrb_12
-> df -v .
Mounted on      Filesystem              Avail/Total      Files      Status
/shared/jrb_12 (DB2HLQ1.DBP0.RSU1407.DB2.ZFS)  107962/144000   4294967214
Available
ZFS, Read/Write, Device:2897, ACLS=Y
File System Owner : SYSD      Automove=Y      Client=N
Filetag : T=off  codeset=0
Aggregate Name : DB2HLQ1.DBP0.RSU1407.DB2.ZFS
wsc1:/shared/jrb_12
->

```

---

SYSA has these characteristics:

- ▶ “which\_dbp0” points at “jrb\_11” and “which\_qual” points at “jrb\_12”.
- ▶ Display of the correct file system on “/shared/jrb\_11” as DB2HLQ1.DBP0.RSU1404.DB2.ZFS
- ▶ Display of the correct file system on “/shared/jrb\_12” as DB2HLQ1.DBP0.RSU1407.DB2.ZFS

Example B-7 shows the environment for the /SYSB file system.

*Example B-7 Environment for wsc2 (SYSB)*

---

```
-> cd /SYSB
wsc2:/SYSB
-> ls -al which*
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 17:05 which_dbp0 -> jrb_11
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 16:58 which_qual -> jrb_12
wsc2:/SYSB
-> cd /shared/jrb_11
wsc2:/shared/jrb_11
-> df -v .
Mounted on      Filesystem              Avail/Total      Files      Status
 /shared/jrb_11 (DB2HLQ1.DBP0.RSU1404.DB2.ZFS)  107962/144000   4294967214
Available
ZFS, Read/Write, Device:2896, ACLS=Y
File System Owner : SYSD      Automove=Y      Client=N
Filetag : T=off  codeset=0
Aggregate Name : DB2HLQ1.DBP0.RSU1404.DB2.ZFS
wsc2:/shared/jrb_11
-> cd /shared/jrb_12
wsc2:/shared/jrb_12
-> df -v .
Mounted on      Filesystem              Avail/Total      Files      Status
 /shared/jrb_12 (DB2HLQ1.DBP0.RSU1407.DB2.ZFS)  107962/144000   4294967214
Available
ZFS, Read/Write, Device:2897, ACLS=Y
File System Owner : SYSD      Automove=Y      Client=N
Filetag : T=off  codeset=0
Aggregate Name : DB2HLQ1.DBP0.RSU1407.DB2.ZFS
wsc2:/shared/jrb_12
```

---

SYSB has these characteristics:

- ▶ “which\_dbp0” points at “jrb\_11” and “which\_qual” points at “jrb\_12”.
- ▶ Display of the correct file system on “/shared/jrb\_11” as DB2HLQ1.DBP0.RSU1404.DB2.ZFS
- ▶ Display of the correct file system on “/shared/jrb\_12” as DB2HLQ1.DBP0.RSU1407.DB2.ZFS

Now run the script “red\_dbp0.sh” on wsc1 as shown in Example B-8.

*Example B-8 DBP1 on wsc1 (SYSA)*

---

```
wsc1:/u/judyrb
-> . ./red_dbp0.sh
RSU1404
wsc1:/u/judyrb
-> echo $STEPLIB
DB2HLQ1.DBP0.SDSNEXIT:DB2HLQ1.DBP0.RSU1404.SDSNLOAD:DB2HLQ1.DBP0.RSU1404.SDSNLOAD2:
none
wsc1:/u/judyrb
```

---

This example shows these items:

- ▶ Output shows **RSU1404**, the current service level on wsc1 (SYSA)
- ▶ The **echo \$STEPLIB** command that displays the correct STEPLIB concatenation

Example B-9 shows the PATH, LIBPATH, and CLASSPATH for this user after the script has run. The only item of interest is that “db2dbp0” is in all the path statements, and that is all that is needed to point to the correct mount point, /shared/jrb\_11 or /shared/jrb\_12.

*Example B-9 Example of PATH, LIBPATH, CLASSPATH for output of script for DBP1*

---

```
-> echo $PATH
/shared/db2dbp0/jdbc/bin:/u/brown/bin:/usr/lpp/Printsrv/bin:/bin:/usr/sbin:/usr/local/bin:/local/tools/bin:/shared/java/J6.0.1/bin
wsc1:/u/judyrb
-> echo $LIBPATH
/shared/db2dbp0/jdbc/lib:/usr/lpp/Printsrv/lib:/lib:/usr/lib:/usr/local/lib
wsc1:/u/judyrb
-> echo $CLASSPATH
/u/judyrb:/shared/db2dbp0/jdbc/classes/db2jcc_license_cisuz.jar:/shared/db2dbp0/jdbc/classes/db2jcc_javax.jar:/shared/db2dbp0/jdbc/classes/sqlj.zip:/shared/db2dbp0/jdbc/classes/db2jcc.jar:/shared/db2a10/base/lib/clp.jar
wsc1:/u/judyrb
->
```

---

Now run “red\_dbp0.sh” on wsc1 (SYSB), as shown in Example B-10.

*Example B-10 DBP1 on wsc2 (SYSB) at current service level (RSU1404)*

---

```
wsc2:/u/judyrb
-> . ./red_dbp0.sh
RSU1404
wsc2:/u/judyrb
-> echo $STEPLIB
DB2HLQ1.DBP0.SDSNEXIT:DB2HLQ1.DBP0.RSU1404.SDSNLOAD:DB2HLQ1.DBP0.RSUDB2HLQ11304.SDSNL0D2:none
wsc2:/u/judyrb
```

---

Its results are the same as those for SYSA, as expected:

- ▶ Output shows **RSU1404**, the current service level on wsc1 (SYSA)
- ▶ The **echo \$STEPLIB** command displays the correct STEPLIB concatenation

Now go to the DB2QUAL group and run its “red\_qual.sh”. It uses the same libraries as WSCDBP0. See Example B-11.

*Example B-11 DB2QUAL on wsc1 (SYSA) at upgraded service level*

---

```
wsc1:/u/judyrb
-> . ./red_qual.sh
RSU1407
wsc1:/u/judyrb
-> echo $STEPLIB
DB2HLQ1.DBP0.SDSNEXIT:DB2HLQ1.DBP0.RSU1407.SDSNLOAD:DB2HLQ1.DBP0.RSU1407.SDSNL0D2:none
wsc1:/u/judyrb
```

---

This example shows these items:

- ▶ Output shows **RSU1407**, the current service level on wsc1 (SYSA)
- ▶ The **echo \$STEPLIB** command displays the correct STEPLIB concatenation

Now run “red\_qual.sh” on wsc2 (SYSB). See Example B-12.

---

*Example B-12 DB2QUAL on wsc2 (SYSB)*

---

```
wsc2:/u/judyrb
-> . ./red_qual.sh
RSU1407
wsc2:/u/judyrb
-> echo $STEPLIB
DB2HLQ1.DBP0.SDSNEXIT:DB2HLQ1.DBP0.RSU1407.SDSNLOAD:DB2HLQ1.DBP0.RSU1407.SDSNLOAD2:
none
wsc2:/u/judyrb
```

---

Both WSCDBP0 and DB2QUAL can now run on each system, SYSA or SYSB, at different levels.

### ***Changing the symlink***

Now upgrade the DBP1 service level from RSU1404 to RSU1407, while keeping DBP2 at level RSU1404. Remove and redefine the symlink “which\_dbp0” to point to /shared/jrb\_l2. The commands to do this are shown in Example B-13.

---

*Example B-13 Changing symlink for new service level used by DBP1 on SYSA*

---

```
:/Users/brown
-> wsc 1

- - - - -
-          IBM Washington Systems Center          -
-          Gaithersburg, Maryland                 -
- - - - -

wsc1:/u/brown
-> cd /SYSA
wsc1:/SYSA
-> ls -al which_dbp0
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 16:57 which_dbp0 -> jrb_l1
wsc1:/SYSA
-> rm which_dbp0
wsc1:/SYSA
-> ln -s jrb_l2 which_dbp0
wsc1:/SYSA
-> ls -al which_*
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 17:05 which_qual -> jrb_l2
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 18:01 which_dbp0 -> jrb_l2
wsc1:/SYSA
-> cd /SYSB
wsc1:/SYSB
-> ll which_*
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 17:05 which_qual -> jrb_l2
lrwxrwxrwx  1 BROWN  SYS1          6 Jul 31 16:58 which_dbp0 -> jrb_l1
wsc1:/SYSB
-> exit
Connection to wsc1.washington.ibm.com closed.
```



```
:/Users/brown  
->
```

---

On SYSA, remove the symlink “which\_dbp0” and define it to point to “jrb\_l1”. Then list the symlinks to show the new settings.

On wsc1 (SYSA), the “/shared/db2dbp0” points to the “/\$SYSNAME/which\_dbp0” symlink, which points to the “jrb\_l2” symlink, which points to the “/shared/jrb\_l2” mount point.

However, on wsc2 (SYSB), “/shared/db2dbp0” still points to the “jrb\_l1”.

### **Test for mixed service levels for DBP1 and DBP2**

Example B-14 shows testing DBP1 to see whether its level has been upgraded.

*Example B-14 Script for wsc1 (DBP1) on SYSA at new service level*

---

```
-> cd /SYSA  
wsc1:/SYSA  
-> df -v which*  
Mounted on      Filesystem                Avail/Total   Files      Status  
/shared/jrb_l2 (DB2HLQ1.DBP0.RSU1407.DB2.ZFS)  107962/144000 4294967214 Available  
ZFS, Read/Write, Device:2897, ACLS=Y  
File System Owner : SYSD      Automove=Y     Client=N  
Filetag : T=off  codeset=0  
Aggregate Name : DB2HLQ1.DBP0.RSU1407.DB2.ZFS  
wsc1:/SYSA  
-> cd /u/judyrb  
wsc1:/u/judyrb  
-> . ./red_dbp0.sh  
RSU1407  
wsc1:/u/judyrb  
-> echo $STEPLIB  
DB2HLQ1.DBP0.SDSNEXIT:DB2HLQ1.DBP0.RSU1407.SDSNLOAD:DB2HLQ1.DBP0.RSU1407.SDSNLOAD2:none  
wsc1:/u/judyrb  
->
```

---

It is correctly at the higher service level:

- ▶ The output of the script is **RSU1407**, the upgraded service level on DBP2, running on SYSA.
- ▶ Display of the correct file system on “/shared/jrb\_l2” as DB2HLQ1.DBP0.RSU1407.DB2.ZFS
- ▶ The **echo \$STEPLIB** command that displays the correct STEPLIB concatenation.

Example B-15 shows running red\_dbp0.sh on SYSB, which uses the lower service level.

*Example B-15 STEPLIB for wsc2 (DBP2) SYSB at older service level*

---

```
-> cd /SYSB  
wsc2:/SYSB  
-> df -v which*  
Mounted on      Filesystem                Avail/Total   Files      Status  
/shared/jrb_l1 (DB2HLQ1.DBP0.RSU1404.DB2.ZFS)  107962/144000 4294967214 Available  
ZFS, Read/Write, Device:2896, ACLS=Y  
File System Owner : SYSD      Automove=Y     Client=N  
Filetag : T=off  codeset=0
```

Aggregate Name : DB2HLQ1.DBP0.RSU1404.DB2.ZFS

```
/shared/jrb_12 (DB2HLQ1.DBP0.RSU1407.DB2.ZFS)    107962/144000    4294967214
Available
ZFS, Read/Write, Device:2897, ACLS=Y
File System Owner : SYSD          Automove=Y          Client=N
Filetag : T=off    codeset=0
Aggregate Name : DB2HLQ1.DBP0.RSU1407.DB2.ZFS
wsc2:/SYSB
-> cd /u/judyrb
wsc2:/u/judyrb
-> . ./red_dbp0.sh
RSU1404
wsc2:/u/judyrb
-> echo $STEPLIB
DB2HLQ1.DBP0.SDSNEXIT:DB2HLQ1.DBP0.RSU1404.SDSNLOAD:DB2HLQ1.DBP0.RSU1404.SDSNLOD2:
none
wsc2:/u/judyrb
->
```

---

Observe that the lower service level is shown as output and also for STEPLIB on SYSB.

The example has shown that DBP1 can operate at the upgraded service level (RSU1407) on SYSA, while DBP2 on SYSB remains at a lower level (RSU1404).

## Summary

This example showed that you can start the IBM Data Service Driver for JDBC and SQLJ, using both type 4 and type 2 connectivity. This configuration provides the flexibility to accomplish these goals:

1. Change *some* DB2s in a system (that is, not all DB2s in a system should point at the same DB2 data sets)
2. Change *some* DB2s in a data sharing group (that is, you do not want to upgrade the whole data sharing group at the same time).
3. Have more than one DB2 subsystem per data sharing group per z/OS system (meaning that you effectively must use DB2 *group attach* to get the full benefit).

You can accomplish this flexibility for z/OS UNIX users as well.

The users need only know one symlink, the high level qualifier of the DB2 libraries, and the group attach name. The latter identifies the data sharing group to access. You do not have to know the system on which you are running, the service level of the DB2 libraries, or which member you are accessing.

**Important:** This example works correctly for those WebSphere for z/OS clients that use type 4 connectivity. It is only necessary to export the PATH.

It does not work for type 2 connectivity. WebSphere uses STEPLIB and for type 2 SDSNLOD2 must exactly match DB2.ZFS. What this example accomplished is to upgrade the service level of DBP1 transparently for the DB2.ZFS, but changes must be made to WebSphere outside of this scenario.

# Related publications

The publications that are listed in this section are suitable for a more detailed description of the topics covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *DFSMSdfs Presentation Guide*, SG24-6973
- ▶ *DFSMSdfs Overview and Planning Guide*, SG24-6971
- ▶ *IBM CICS Performance Series: CICS and VSAM RLS*, REDP-4905
- ▶ *New Ways of Running Batch Applications on z/OS: Volume 1 CICS Transaction Server*, SG24-7991
- ▶ *Partitioned Data Set Extended Usage Guide*, SG24-6106
- ▶ *z/OS Planned Outage Avoidance Checklist*, SG24-7328
- ▶ *z/OS Version 1 Release 8 Implementation*, SG24-7265
- ▶ *z/OS Version 1 Release 9 Implementation*, SG24-7427
- ▶ *z/OS Version 1 Release 10 Implementation*, SG24-7605
- ▶ *z/OS Version 1 Release 11 Implementation*, SG24-7729
- ▶ *z/OS Version 1 Release 12 Implementation*, SG24-7853
- ▶ *z/OS Version 1 Release 13 Implementation*, SG24-7946
- ▶ *z/OS Version 2 Release 1 Implementation*, SG24-xxxx

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

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

## Other publications

These publications are also relevant as further information sources:

- ▶ *CICS Batch Application Control for z/OS Version 1 Release 1 Modification 1 User's Guide*, SC34-6321
- ▶ *CICS Batch Application Control for z/OS Version 1 Release 1 Workstation User's Guide*, SC34-6322
- ▶ *CICS Interdependency Analyzer for z/OS Version 5 Release 1 User's Guide and Reference*, SC34-2811
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1: What's New*, GC34-2879
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 Application Programming Guide*, SC34-2844

- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 Application Programming Reference*, SC34-2845
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 Customization Guide*, SC34-2847
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 Operations and Utilities Guide*, SC34-2863
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 Problem Determination Guide*, SC34-2865
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 RACF Security Guide*, SC34-2866
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 Resource Definition Guide*, SC34-2848
- ▶ *CICS Transaction Server for z/OS Version 5 Release 1 System Programming Reference*, SC34-2872
- ▶ *CICS Transaction Server for z/OS Version 4 Release 2: What's New*, GC34-7192
- ▶ *CICS Transaction Server for z/OS Version 4 Release 1: What's New*, GC34-6994
- ▶ *CICS VSAM Transparency for z/OS Version 2 Release 1 User's Guide*, SC34-7249
- ▶ *z/OS Communications Server: New Functions Summary*, GC27-3664
- ▶ *z/OS DFSMS Using the New Functions*, SC26-7473
- ▶ *z/OS Introduction and Release Guide*, GA32-0887
- ▶ *z/OS Migration*, GA22-7499
- ▶ *z/OS MVS Initialization and Tuning Reference*, SA23-1380
- ▶ *z/OS MVS System Commands*, SA38-0666
- ▶ *z/OS Planning for Installation*, GA32-0890
- ▶ *z/OS Summary of Message and Interface Changes*, SA22-7505

## Online resources

These websites are also relevant as further information sources:

- ▶ CICS Support pack: CA1Y "Send email from CICS Transaction Server for z/OS"  
<http://www.ibm.com/support/docview.wss?uid=swg24033197>

## Help from IBM

IBM Support and downloads

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

IBM Global Services

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



## Improving z/OS Application Availability by Managing Planned Outages

(0.5" spine)  
0.475" <-> 0.873"  
250 <-> 459 pages







# Improving z/OS Application Availability by Managing Planned Outages

**Provides a definition of a planned outage**

**Describes various types of planned outages**

**Covers z/OS and its major subsystems**

This IBM Redbooks publication is intended to make System Programmers, Operators, and Availability Managers aware of the enhancements to recent releases of IBM z/OS and its major subsystems in the area of planned outage avoidance. It is a follow-on to, rather than a replacement for, z/OS Planned Outage Avoidance Checklist, SG24-7328.

Its primary objective is to bring together in one place information that is already available, but widely dispersed. It also presents a different perspective on planned outage avoidance. Most businesses care about application availability rather than the availability of a specific system. Also, a planned outage is not necessarily a bad thing, if it does not affect application availability. In fact, running for too long without an IPL or subsystem restart might have a negative impact on application availability because it impacts your ability to apply preventive service. Therefore, this book places more focus on decoupling the ability to make changes and updates to your system from IPLing or restarting your systems.

## 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)

SG24-8178-00

ISBN 0738440213