

Rocket Model 204 System Manager's Guide

Version 7 Release 4.0

May 2012
204-0704-SM-01



Notices

Edition

Publication date: May 2012

Book number: 204-0704-SM-01

Product version: Rocket Model 204 System Manager's Guide Version 7 Release 4.0

Copyright

© Computer Corporation of America 1989-2012. All Rights Reserved.

Computer Corporation of America is a wholly-owned subsidiary of Rocket Software, Inc.

Trademarks

Rocket is a registered trademark of Rocket Software, Inc. For a list of Rocket registered trademarks go to: www.rocketsoftware.com/about/legal. All other products or services mentioned in this document may be covered by the trademarks, service marks, or product names of their respective owners.

License agreement

This software and the associated documentation are proprietary and confidential to Rocket Software, Inc., are furnished under license, and may be used and copied only in accordance with the terms of such license.

Note

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulation should be followed when exporting this product.

Contact information

Web Site: www.rocketsoftware.com

Rocket Software, Inc. Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA
Tel: +1.617.614.4321
Fax: +1.617.630.7100

Contacting Technical Support

If you have current support and maintenance agreements with Rocket Software and CCA, contact Rocket Software Technical support by email or by telephone:

Email: m204support@rocketsoftware.com

Telephone :

North America +1.800.755.4222

United Kingdom/Europe +44 (0) 20 8867 6153

Alternatively, you can access the Rocket Customer Portal and report a problem, download an update, or read answers to FAQs. You will be prompted to log in with the credentials supplied as part of your product maintenance agreement.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

Contents

Notices
Contacting Technical Support

About this Guide

Audience.....	XXV
Model 204 documentation set	XXV
Documentation conventions	XXV

I Model 204 Operational Issues

1 Model 204 Configurations and Operating Environments

In this chapter	3
Overview	3
About Model 204	4
Staffing recommendations	4
Model 204 configurations	5
Operating systems	5
Operating environments	5
CPU	6
Direct Access Storage Devices (DASD)	6
Shared DASD and Global Resource Serialization	6
ENQCTL command enhanced	7
Cautions using Global Resource Serialization	7
64-bit architecture support	7
Understanding above the bar storage	8
Set the IBM MEMLIMIT system option	8
Model 204 entities in above the bar storage	8
Handling 64-bit statistics	9
Model 204 storage	9
z/VSE storage considerations	9
CMS disk considerations	10
Related Rocket Software products	11

2 Defining the Runtime Environment (CCAIN)

In this chapter	13
Overview	14
Structure of CCAIN	14
ONLINE data streams with CCAIN	15
Parameter lines	15
z/OS JCL	15
z/VSE JCL	16
CMS CCAIN file	17
z/VM ONLINE processing	18
EXEC procedure defining the ONLINE environment	18
CMS ONLINE command	18
CMS utilities and EXECs	20
Using the M204UTIL utility	21
Using the M204MOUN EXEC to mount and dismount tapes	22

BATCH204 JCL with CCAIN.....	23
SAMPLE: z/OS JCL for invoking a BATCH204 run	23
Runtime environment specifications	23
Specifying EXEC statement parameters.....	23
Calculating region size	24
Setting the SYSOPT parameter.....	25
SYSIPT logical unit in z/VSE.....	25
User 0 input file in the z/VM environment	26
Stacking z/VM runtime parameters	26
User 0 parameters	26
z/VSE UPSI Job Control statement	30
Resource locking	31
Record locking table.....	32
Resource locking table	32
Multiple jobs running on one CPU.....	33
Handling locking conflicts.....	33
Data integrity.....	34
Multiple Model 204 versions running on separate CPUs	34
How Model 204 resolves locking conflicts.....	35
Locking conflicts between CPUs.....	35
Shared DASD locking conflicts	36
z/VSE considerations.....	36
Resource locking in z/VM	37
Disk buffers and Model 204 storage	37
Buffers in an ONLINE configuration	38
Using 31-bit storage	39
Using 64-bit storage	39
Managing above the bar storage.....	40
HASHCELL - Allocation of hash cells per buffer pool page	42
Monitoring disk buffers.....	43
Managing delayed disk updates.....	43
Specifying delayed updates—DKUPDTWT parameter.....	43
Handling delayed updates and CHKPPST	44
Attempting a checkpoint	44
Factors affecting disk update and checkpoint processing	45
Understanding file statistics	45
Handling 64-bit statistics.....	46
Server areas.....	46
Sizing user server areas	46
Initialization and error handling	47
Calculating fixed table size	47
Calculating variable table size	48
Server tables	50
Full-screen buffer table (FSCB)	50
File group table (FTBL).....	51
Understanding the global variable table (GTBL)	52
Dummy string and \$READ table (ITBL)	53
Labels, names, and variables table (NTBL)	53
Internal statement table (QTBL).....	54
QTBL and IFAM processing.....	54
QTBL and FLOD processing	54
Sample QTBL entries	55
User, field, group security table (RTBL).....	57
Character string table (STBL).....	58

Temporary work page list table (TTBL)	59
Compiler variable table (VTBL)	59
Procedure security table (XTBL)	61
3 Defining the User Environment (CCAIN)	
In this chapter	63
Overview	64
Setting user parameters	64
Basic rules	64
Access methods and device types	64
Access methods	65
Device types (IODEV settings)	65
BSAM (IODEV=3)	67
Uses of IODEV=3	68
Input or output data set definitions	69
Buffer allocation	71
Dynamic allocation and DFSMS/HSM	71
SNA Communications Server terminal support (IODEV=7, 37)	72
SNA Communications Server network definition requirements	72
IP address using TN3270 connection to VTAM session: IPADDR	73
CCAIN requirements	73
CMS/SNA Communications Server Interface for full-screen terminals	74
CRAM (IODEV=11, 23, 29)	74
Facilities requiring CRAM	75
CRAM options for z/OS	75
Invoking a CRAM option	77
Implementing CRAM XDM usage for z/OS operating systems	77
SNAPCRAM utility	78
Cross-memory debugging facility	78
Monitoring XDM	79
Working with CRAM	82
Using CRAM on a z/OS operating system	84
Using CRAM on a z/VSE operating system	87
CRAM IFSTRT protocol IFAM2 (IODEV=23) requirements	92
Multiple Online copies on a single system	92
TSO Interface	93
INTERCOMM Interface	94
CICS Interface	94
SQL server threads (IODEV=19)	94
Coding the IODEV=19 line	94
Example for z/VM	95
Horizon (IODEV=27)	96
Managing IODEV=27 threads	97
IUCV (IODEV=39, 41, 43)	97
Programs required for the IUCV CMS Interface	98
Interactive access (M204USR MODULE)	99
Line-at-a-time (IODEV=39)	99
IUCV full-screen (IODEV=41)	100
IFAM2 (IODEV=43)	100
RCL thread (IODEV=49)	100
M204 command (CMS)	101
M204 syntax	101
Command File facility	103
Command file processing	103

CMS service machine console (ALTIODEV=45, 47)	105
Setting the ALTIODEV parameter	105
Executing in single-user mode	105
User environment control parameters	105
4 Controlling System Operations (CCAIN)	
In this chapter	111
Overview	111
System control commands	112
ONLINE termination	117
Manual termination	117
Time-out termination	118
Operator control	118
HALT processing	119
EOD processing	119
ONLINE monitoring	120
Unformatted displays	120
Formatted displays	120
User information in formatted displays	121
MONITOR examples	125
MONITOR LINK example	126
MONITOR PROCESSGROUP example	128
MONITOR PROCESS example	130
MONITOR DISKBUFF example	131
Console operator communications with Model 204 (z/VSE)	131
Pseudo subtasks	132
Statistics and monitoring of pseudo subtasks	133
Messages	133
Job step return codes	134
Using MSGCTL command with SYSOPT parameter to produce an abend	134
Conditional job control	135
Priority scheduling	135
CUSTOM=(9) setting	140
Scheduler performance parameters	140
Dynamic dispatching	141
Dynamic dispatching rules	142
Balancing CPU bound and I/O bound users	142
Queue aging	143
User 0's priority	143
Capturing abends	143
5 Using the System Scratch File (CCATEMP)	
In this chapter	145
Overview	145
Sizing CCATEMP	146
Transaction back out in recovery	146
FLOD exits	147
File groups	147
Subsystem Management and precompiled requests	147
CCATEMP in system recovery	148
Shared DASD	148
After a system crash	148
Using cache fast write	149
CCATEMP statistics	149
Handling 64-bit statistics	149

CCATEMP statistics at termination	150
z/OS considerations	150
Storing record set bit maps	150
CCATEMP in Storage feature	151
Activating the CCATEMP in Storage feature for a job	151
System programmers consideration.....	151
Tracking dataspaces and hiperspaces	152
z/VSE and CCATEMP considerations.....	152
Allocating CCATEMP	152
Label information.....	153
Multiple CCATEMP data sets	153
6 Storing and Using File Group Definitions (CCAGRP)	
In this chapter	155
Overview	155
File groups.....	156
Types of file groups.....	156
Permanent groups	156
Temporary groups	156
Ad hoc groups	157
Using file groups	157
Accessing cyclic data.....	157
Archiving aging data.....	157
Accessing independent but similar data	158
Referencing new file names without changing application programs	158
Updating File Groups.....	158
Creating the CCAGRP data set	158
Sizing CCAGRP	158
z/OS procedures	159
z/OS JCL example	159
z/VSE procedures.....	160
z/VM procedures.....	160
Storing group definitions in CCAGRP.....	161
Creating file groups	161
Examples.....	162
Opening, closing, displaying, and deleting file groups	162
Opening and closing file groups	162
Displaying file group information	163
Deleting file groups.....	163
File group passwords and privileges.....	163
File group passwords.....	163
File group privileges.....	164
7 Creating Server Data Sets (CCASERVER)	
In this chapter	165
Overview	165
Server swapping and the CCASERVER file.....	166
CCASERVER in Storage feature.....	166
System programmers consideration.....	167
Tracking dataspaces and hiperspaces	167
Activating the CCASERVER in Storage feature for a job.....	167
Sizing and storage considerations	167
Physical devices and server size	167
Calculating CKD disk storage requirements	168
Calculating FBA storage requirements	168

Disk storage calculation examples.....	169
Requirements for server swapping	169
Parameters	169
Multiple server data sets.....	169
Server data sets on CKD devices	170
ECKD channel program support	170
Using cache fast write	171
Allocation and job control	171
z/OS	171
z/VSE.....	171
FTBL and above the bar storage	172
Parameters	172
Description	172
Usage notes.....	172
Messages	172
8 System Requirements for Application Subsystems	175
In this chapter	175
Overview of CCASYS.....	176
Maintaining CCASYS.....	176
Creating CCASYS	176
Reorganizing CCASYS	177
Using CCASYS	177
Subsystems available at initialization	177
JCL requirements.....	177
File security.....	177
Maintaining CCASYS	178
Recovering CCASYS.....	178
Activating the APSY Precompiled Procedures In Storage feature.....	178
Storage improvements for APSY subsystem saved precompilation	179
Storage requirements.....	179
Setting the DSPOPT parameter	180
APSY load statistics	180
Parallel Query Option/204 and scattered subsystems.....	181
Overview of the Subsystem Management facility.....	181
Components of a subsystem.....	182
File and group availability	183
Member availability to APSY subsystems.....	183
Member availability to subsystem users.....	183
Enabling a disabled subsystem file	184
Disabling a subsystem file.....	184
Application subsystem processing	185
Subsystem startup	185
Login processing	185
Main processing.....	186
Requirements for using temporary groups	187
Disconnect processing	187
Error processing	188
Subsystem operating requirements	188
Required files	189
Resource locking table space.....	189
Minimum server table sizes	190
CCATEMP space	190
SPCORE size	191

Subsystem operating options.....	191
Subsystem commands	195
AUTOSYS parameter	196
Overview of the SUBSYSMGMT interface	196
SUBSYSMGMT facility screen summary.....	197
Using secondary screens	197
Sample screens and PQO.....	198
SUBSYSMGMT Activity screen.....	198
Commands and options	198
Overview of activities.....	199
From fields (PQO only).....	199
Revising command privileges.....	199
SUBSYSMGMT Privileges screen.....	200
Entering changes in the COMMAND PRIVILEGES screen.....	201
Operational Parameters screen	202
Commands.....	203
Parameter descriptions	203
Message display options	205
Procedure Specifications screen	205
Commands.....	206
Procedure prefixes	206
Procedure names.....	206
Global variables.....	207
Subsystem File Use screen	208
Commands.....	209
File specifications	209
Subsystem Classes screen	210
Commands.....	211
SUBSYSTEM NAME and CLASS.....	212
Security specifications	212
User Definitions screen.....	213
Commands.....	213
Screen specifications.....	214
Subsystem Class Users screen.....	214
COPY SCLASS USERID option	215
Sample Preview screen	215
Subsystem Import/Export options	216
Export.....	217
Import.....	217
Export Delete	218
Exportlist.....	218
Exportable data: scope and limitations.....	218
Subsystem Administration screen.....	218
Admin options.....	219
Subsystem Pattern field	220
Administrative Privileges screen.....	220
Hierarchy of privileges.....	221
PF keys	221
ADDPRIV SCLASS and Add Privileges.....	222
Summary of subsystem privileges.....	222
User Matrix screen	223
Dynamic APSY support.....	224
Introducing dynamic APSY subsystem attribute changes	224
Dynamically refreshing procedure compilation with the REFRESH SUBSYSPROC command	228

Using precompilable procedures with commands	234
Handling subsystem error procedures	234
9 Customizing Functions and Translation Tables	
In this chapter	235
Overview	235
Adding functions to the FUNU module.....	236
Coding a function	237
Coding conventions	237
String arguments.....	237
Numeric arguments.....	238
Omitted arguments.....	238
Using the ENTER macro to allocate working storage.....	239
Return values	240
Coding messages	240
Encoding/Decoding facility	241
CDTB module.....	241
Customizing the sample translation tables	242
Modifying translation tables	242
Sizing the CDTB module	243
Converting user-written functions.....	243
Coding requirements for all operating systems.....	243
Additional requirements for systems using 31-bit architecture	244
Additional requirements for MP/204 (multiprocessing)	245
 II Managing Security	
10 Storing Security Information (CCASTAT)	
In this chapter	249
Overview	249
Creating CCASTAT	249
z/OS procedures	250
z/VM procedures.....	250
z/VSE procedures.....	250
Using CCASTAT.....	251
Data set definitions (z/OS, z/VSE, z/VM)	251
z/VSE security considerations	251
Login security.....	252
Login implementation.....	252
Login delays	253
Password table data set	253
Password table maintenance.....	254
Backing up the password table	254
Adding entries	255
Deleting entries.....	255
Changing entries	255
A sample dialogue using LOGCTL	256
Processing login password table updates.....	256
Updates that exceed allocated storage	256
Multiple copies of Model 204.....	257
Overview of the Password Expiration feature	258
Tracking the days a password is valid.....	258
Revoking passwords and suspending user IDs	258
Deleting CCASTAT entries	259

Defining a password.....	259
FILE and GROUP passwords.....	259
Create a backup copy of CCASTAT.....	259
Capturing diagnostic data	260
Increase in CCASTAT	260
Understanding the ZCTLTAB utility	260
Updating CCASTAT using the Password Expiration feature	260
Setting the security parameters.....	260
Displaying the security parameters	261
Changing the values of the security parameters.....	261
ZCTLTAB condition codes	261
Running ZCTLTAB to update CCASTAT	262
System manager's responsibilities	265
Supporting password expiration at your site.....	266
Deleting a user ID	266
Defining passwords.....	266
Changing and reusing passwords.....	266
Password from a trusted environment.....	267
CCASTAT backwards compatibility.....	267
Managing CCASTAT messages	267
11 Establishing and Maintaining Security	
In this chapter	269
Overview	269
File security	270
OPENCTL parameter.....	270
PRIVDEF parameter.....	270
PRIVDEF parameter settings.....	270
File password table maintenance	271
Using LOGCTL to modify the file password table	271
Sample dialogue using LOGCTL.....	272
Adding and removing security from files	272
Group security	273
LOGCTL and LOGGRP commands	273
Record security	273
Field-level security	274
Scope of field-level security	274
Field access types	275
Sample dialogue using LOGCTL.....	275
Procedure security	276
Sample dialogue using LOGCTL.....	276
Terminal security	276
III Auditing and Problem Determination	
12 Obtaining User 0 Output (CCAPRINT)	
In this chapter	281
Overview	281
CCAPRINT data set.....	281
Initialization information about Early Warnings	282
z/OS considerations.....	282
z/VSE considerations	282
z/VM considerations	282

13 Tracking System Activity (CCAJRNL, CCAAUDIT, CCAJLOG)

In this chapter	285
Overview of the journal data sets	286
Introducing CCAJRNL.....	286
Introducing CCAAUDIT and CCAJLOG.....	286
Version-specific journals.....	287
Using Model 204 journal files.....	287
Creating and generating CCAJRNL	287
CCAJRNL as single data set or stream	288
SWITCH STREAM command.....	288
Using the SWITCH STREAM command	288
Journal block header information for SWITCH STREAM.....	289
Using a SWITCH STREAM CCAJRNL command during extended quiesce	289
SWITCH STREAM limitations.....	290
SWITCH STREAM command for concatenated streams	291
Handling streams without records	291
Recovery parameters	292
Writing error messages to the journal data set.....	292
ERRMSG - Setting the length of saved error messages	292
Using the CCAJRNL data set	293
Performance efficiencies using CCAJRNL	293
Choosing whether to define CCAAUDIT	293
Allocating multiple journal buffers.....	294
Using the CCAJLOG data set	294
M204JLOG assembler exit	295
Model 204 roll forward recovery with CCAJLOG	297
Separating transaction (CCAJRNL) and auditing (CCAJLOG) information	297
Regenerating files using CCAJRNL and CCAJLOG	297
Considerations for CCAJLOG.....	297
Using the journals correctly.....	297
CCAJRNL data set record layout.....	298
Journal entry format.....	298
Number of lines in the journal data set.....	299
Sizing the journal buffer	300
Managing space requirements for all operating systems	301
Introducing Model 204 statistics.....	301
Audit trail and journal statistics lines	301
Output lines.....	301
Identifying subtypes	302
Monitoring statistics.....	302
Setting the NSUBTKS parameter	302
Setting parameters to collect certain statistics	303
Audit trail format	303
Generating an audit trail.....	305
Overview of audit trail parameters	305
z/VSE and the audit trail.....	305
z/VM and the audit trail.....	306
Introducing the AUDIT204 utility.....	306
Input to AUDIT204	307
ANALYZE command	308
FORMAT command.....	309
REPORT command	310
Using the AUDIT204 utility.....	312
CCAJRNL and CCAJLOG data sets as input to utilities	312

Journal stream configurations.....	313
z/OS JCL for AUDIT204	313
Samples of a z/OS job stream	314
z/VSE JCL for AUDIT204.....	316
Sample z/VSE job stream	316
(z/VM) CMS JCL for AUDIT204.....	317
Searching multiple tapes for journal and off load data.....	318
14 Storing Diagnostic Information (CCASNAP and CCAMDMP)	
In this chapter	319
Overview	319
SNAPCTL parameter	320
Handling error messages with MSGCTL command	320
Introducing message types	321
Adjusting error messages at your site	321
Understanding MSGCTL command options	322
Using the SNAPFAIL and SNAPFLIM parameters	323
Operating system requirements	324
z/OS	324
z/VSE.....	324
z/VM	324
Using unformatted system dumps (z/OS)	325
Allocating SYSMDUMP data sets	325
CCAMDMP data set DCB characteristics	326
Processing CCAMDMP data sets	327
IV Recovery	
15 Checkpoints: Storing Before-Images of Changed Pages	
In this chapter	331
Overview of the Checkpoint facility.....	331
Understanding the Checkpoint facility.....	332
Checkpoint parameters.....	333
Taking a checkpoint	333
Checkpoint algorithm.....	334
Taking asynchronous checkpoints (31-bit).....	334
Using the CHECKPOINT command	334
Checkpoint processing with IFSTRT or IFFNSH calls.....	335
Determining the status of a checkpoint	335
Aborting a checkpoint.....	335
M204CKPX checkpoint user exit	335
Overview of sub-transaction checkpoints.....	337
Reviewing transaction checkpoints.....	337
Introducing sub-transaction checkpoints	337
Using sub-transaction checkpoints in recovery	338
Using transaction or sub-transaction checkpoints	338
Considerations for implementing sub-transaction checkpoints	339
Implementing sub-transaction checkpoints in your job.....	339
Requirements for CPTYPE=1	339
Checkpoint definition restrictions for sub-transaction checkpoints	340
Before-image logging.....	340
Creating the CHKPOINT/CHKPNT (and CHKPNTS) data set.....	340
Maintaining single volume CHKPOINT data sets	342
Limiting the size of CHKPOINT	342

Obtaining checkpoint information (UTILC).....	342
Checkpoint record formats.....	343
UTILC input data sets	344
UTILC options.....	344
UTILC examples.....	345
16 System and Media Recovery	
In this chapter	347
Overview	348
Consulting other Model 204 manuals	349
Preparing for media recovery	349
Recovery features	350
Handling error diagnosis	350
Handling Transaction Back Out.....	350
Handling a system recovery	351
Handling a media recovery.....	351
Understanding transaction back out.....	352
Making updates permanent.....	352
Transaction boundaries	352
Updates that can be backed out.....	353
Updates that cannot be backed out.....	353
Back out mechanism	354
Lock-pending-updates locking mechanism.....	354
Back out and constraint logs.....	354
Understanding the back out log.....	355
Understanding the constraints log.....	355
Optimizing the constraints log performance.....	355
Disabling transaction back out.....	356
Disadvantages to disabling transaction back out	356
ROLL BACK facility	357
ROLL BACK processing	357
ROLL FORWARD facility	359
RESTART ROLL FORWARD recovery is version specific.....	359
Logging ROLL FORWARD processing	360
Message for the Operator	360
ROLL FORWARD processing	360
ROLL FORWARD file types	361
Journal data set in ROLL FORWARD logging.....	362
File discontinuities	362
Understanding update units.....	363
Determining the starting point for ROLL FORWARD processing.....	364
ROLL FORWARD processing and user hard restarts	364
Back outs during ROLL FORWARD processing	365
Logging ROLL FORWARD processing	365
RESTART recovery considerations for sub-transaction checkpoints.....	366
Considering Roll Back and Roll Forward	366
For RESTART ROLL BACK only processing	366
Restarting after a system failure.....	367
Reporting recovery status	367
Reporting facilities	367
Reporting ROLL FORWARD processing.....	368
Final status of files affected by RESTART recovery	368
Opening files and their status reports	369
Suppressing status reporting.....	369

Requesting status reports	369
Recovery data sets and job control	371
Data sets required for RESTART recovery	371
Sample z/OS JCL for a recovery run	372
Sample z/VSE JCL for recovery	372
z/VSE example 1: checkpoint and journal files on disk (FBA)	373
z/VSE example 2: checkpoint and journal files on tape	373
z/VSE example 3: recovery-restart from disk	374
z/VSE example 4: recovery-restart from tape	375
Statements required for z/VM recovery	375
z/VM recovery examples	376
Handling recovery failures	377
Determining the cause of the error	377
Correcting errors	377
Automated secondary recovery: reuse JCL	377
Sizing the checkpoint data sets correctly	378
Rerunning RESTART recovery after a successful recovery	378
ROLL BACK processing	379
Operational changes to ROLL FORWARD processing	379
RESTART recovery requirements - CCATMP, NFILES, NDIR	381
Recovering deferred update mode files	381
How recovery works for deferred update files	382
Deferred update recovery not supported under z/VSE	382
Recovering dynamically allocated data sets	382
How recovery works for dynamically allocated data sets	382
Bypassing dynamic file allocation	382
Media recovery	382
Phases of a media recovery run	383
Input journal processing for the REGENERATE command	383
Input journal processing for the REGENERATE ONEPASS command	383
Running REGENERATE with the IGNORE argument	384
Number of files that can be regenerated	385
Using the REGENERATE or the REGENERATE ONEPASS command	385
Required data sets for media recovery	386
Example: z/OS media recovery run	387
Media recovery NonStop/204	388
Managing third-party backups	388
Programming a third-party backup	389
Programming a wait for extended quiesce	390
Submitting a backup	391
Managing the extended quiesce	392
Delayed ending for extended quiesce or status message	392
Using the UTILJ utility	393
UTILJ and stream configurations	393
UTILJ options	394
UTILJ REPORT options	396
UTILJ return codes	398
z/VSE considerations	398
UTILJ examples	399
Using UTILJ to analyze problems	400
Defining CCAOUT for additional diagnostics	400
Using the MERGEJ utility	401
Merging overlapping journal files	401
MERGEJ support for journal types	401

MERGEJ considerations	404
Data sets required for the MERGEJ utility	404
z/OS MERGEJ example	405
z/VSE MERGEJ example	405
Using MERGEJ in z/VSE	406
Using MERGEJ in CMS	406
Using MERGEJ with GDG streams	407
17 Configuring Checkpoint and Journal Data Streams	
In this chapter	409
Overview	409
DEFINE STREAM command	410
Using the DEFINE STREAM command	410
Off load and control streams overview	411
Ring stream configuration	411
Output ring stream processing sequence	412
Types of off load streams	412
Off loading and the CLOSE option	412
Off loading and system termination	413
I/O error detection	413
Off loading ring streams from disk to tape	413
Ring streams and recovery	414
Input ring streams (recovery)	414
I/O error processing for input ring streams	415
Ring streams as input to RESTART recovery	415
Concatenated stream configuration	416
Processing sequence	416
Parallel stream configuration	417
Checkpoint configuration support	417
Parallel stream open processing	417
Output parallel streams processing	418
Output error processing	418
Input parallel stream processing	419
Perpetual journaling for z/OS	419
Using the GDG option	420
Data Set Control Blocks (DSCBs) and GDGs	420
Example 1: Ring/parallel journal stream	421
DEFINE STREAM example	422
z/OS JCL	423
z/VSE JCL	423
CMS FILEDEFS	424
Example 2: Parallel checkpoint stream	424
DEFINE STREAM command for parallel checkpoint stream	424
Recovery processing	425
z/OS JCL	425
z/VSE JCL	425
CMS FILEDEFS	426
Example 3: Concatenated tape off load stream for z/OS	426
DEFINE STREAM commands	426
JCL for concatenated offload	427
Troubleshooting GDG streams using specific generations	427

V Performance

18 Performance Monitoring and Tuning

In this chapter	431
Overview	432
Optimizing operating system resources	432
z/OS	432
z/VM	433
z/VSE.....	433
Timer PC	433
EXCPVR	433
IOS BRANCH ENTRY	434
Activating IOS BRANCH ENTRY	435
Saving storage when using IOS Branch Entry	435
Disk buffer monitor statistics and parameters.....	435
DKBM efficiency	435
DKRD and DKPR statistics	435
DKRR statistic	436
Adequacy of existing buffers	436
Anticipatory writes.....	437
Disk update file statistics	439
Page fixing.....	439
1MB virtual pages	441
Parameters	441
Offloading Model 204 work to zIIP processors	443
Model 204 workload eligible for zIIP offload.....	443
Model 204 parameters for zIIP support.....	444
Balancing channel load	448
Reducing storage requirements	449
Dynamic storage allocation tracking.....	449
VIEW statistics	449
User interfaces	450
Types of storage acquisition	450
Cache fast write for CCATEMP and CCASERV	450
Sequential processing.....	451
Prefetch feature	451
How to use Prefetch	451
Long requests	452
IFAM2 control.....	452
SQL system and user statistics	453
SQL statement statistics.....	453
HEAP and PDL high-water marks	454
Interpreting RECDS and STRECDs statistics.....	454
Interpreting the PBRsFLT statistic	454
Interpreting the SQLI and SQLO statistics	455
z/OS recovery	455
Resident Request feature for precompiled procedures	456
Performance considerations	456
Storage protection for z/OS.....	457
System parameters	457
SVPAGES statistic	457
Analyzing MONITOR command output	458
Multiprocessing (MP/204)	458
CPU accounting	459
Activating MP.....	459
MP performance and tuning	460

MP statistics	460
Analyzing CPU utilization (CPU, STCPU, and STDEQ)	462
Analyzing throughput	462
STDEQ and STCPU	463
MP overhead	463
User Language considerations	464
Scheduler operation and accounting (SCHDOPT)	465
MP User Language statement processing	467
Declarative statements	467
FIND statement	468
SORT statement	468
Mixed mode statements	468
Change in User Language READ SCREEN processing	469
Serial statements	469
\$Functions	469
MP program constructs	470
Main part of the request	470
Loops	470
Subroutines	471
ON UNITS	471
Optimization using MPOPTS and MP OPTIONS	471
MP OPTIONS/MPOPTS keywords and parameter settings	472
Loop	473
Calls	473
Functions	473
Mixed mode statements	474
Special keywords	474
Tuning techniques: an example	474
Pros and cons of tuning User Language requests for MP	475

VI System-level Capabilities Available in Model 204

19 Using HLI and Batch Configurations

In this chapter	479
Overview	479
IFAM 1	480
z/OS and IFAM1	480
z/VSE partition usage	481
Running IFAM1 under z/VSE	482
IFAM2	482
CRAM requirements for z/OS and z/VSE	483
Length of HLI functions	483
IFAM buffer size (IFAMBS)	483
Setting the IFAM2 channel name	484
Additional modules (IFAM, CRIO, IFFII, IFIF)	484
Multiple IFAM2 versions	484
Terminating IFAM2 processing	485
z/VSE and IFAM2	485
z/VM and IFAM2	486
IFAM4 for z/OS	486
Link-editing requirements	486
IFAM4 running JCL	488
Common IFAM4 problems	489

BATCH204.....	489
BATCH2 utility.....	490
Managing DCBLRECL to avoid an ABEND	490
20 Allocating and Directing Files Dynamically	
In this chapter	491
Overview	492
Dynamic file allocation	492
DEFINE DATASET command	493
z/OS considerations.....	493
Large data set support	494
z/VM considerations.....	495
Uses of DEFINE DATASET in the z/VSE environment	495
z/VSE DEFINE DATASET options.....	496
z/VSE considerations	496
Extended Task Input/Output Table support.....	497
Using TIOT and XTIO with data sets	497
Using TIOT and XTIO with dynamically allocated Model 204 files	497
Special considerations for obsolete form of ALLOCATE	498
ALLOCATE and FREE commands (z/OS and z/VM)	498
ALLOCATE command.....	499
FREE command	500
Directed output.....	501
Command Overview	501
USE command (z/OS, CMS, guest under z/VM)	502
DEFINE PRINTER (z/OS, CMS, guest under z/VM)	502
z/OS considerations.....	502
CMS considerations	503
DEFINE PUNCH (z/OS, CMS, guest under z/VM)	503
z/VSE output data sets	503
USE PRINTER command (z/VSE under z/VM)	504
USE PUNCH command (z/VSE).....	505
Example 1: USE PUNCH in z/VSE	505
Example 2: USE PUNCH in z/VSE running as guest under z/VM	506
Using z/VSE/POWER.....	507
Printer and punch devices	507
z/VSE/POWER JCL example.....	507
z/VSE/POWER system requirements: the \$\$\$BSGMNT transient	508
z/VSE/POWER job stream	508
Tailoring the Model 204 separator page.....	509
21 Accessing BSAM and VSAM Files	
In this chapter	511
Overview	511
Sequential I/O processing	512
Buffer requirements	512
Determining sequential data set format	512
Sequential data set algorithm	513
Examples of sequential data set processing	515
VSAM I/O processing	515
System requirements.....	516
Primary key processing	516
Alternate index key processing	516
Loading a VSAM KSDS or ESDS file	517

22	Model 204 External Call Facility	
	In this chapter	519
	Overview	519
	Working with ECF	520
	Loading an external module	520
	Calling an external module.....	520
	Avoiding data corruption and incorrect results	520
	Stopping an external load module	521
	ECF statistics and messages	521
	BUMP command enhanced with the FORCE option for ECF users	521
	ECF User Language statements.....	522
	EXTERNAL CALL statement.....	522
	EXTERNAL DELETE statement	523
	EXTERNAL LOAD statement	523
	EXTERNAL MODULE statement.....	523
	EXTERNAL NAME statement.....	525
	EXTERNAL START statement	525
	EXTERNAL STOP statement	526
	ECF return codes and \$function.....	526
	\$ECFSTAT function.....	528
	ECF User 0 parameters	528
	Subtask and load module management.....	529
	Subtasks assignment	529
	Subtask affinity.....	529
	Load modules	530
	Fulfillment order	530
	Restrictions and cautions.....	530
	Tracking ECF	531
	Wait types for ECF	531
	ECF statistics	532
	ECF examples	532
	COBOL sample Number 1.....	532
	COBOL sample Number 2.....	534
	SAS/C sample.....	536
	Assembler sample	537
23	Using Program Communication Facilities	
	In this chapter	541
	Overview	541
	Preparing for TPROCESS communication.....	542
	Defining the partners.....	542
	Preparing for CICS TPROCESS	543
	Preparing for CMS TPROCESS	543
	For more information.....	544
	Preparing for Transfer Control communication	544
	SNA Communications Server transfers are more complex	544
	Defining the partners.....	545
	Transferring to CICS.....	545
	Transferring to SNA Communications Server	545
	For more information.....	546
A	Using System Statistics	
	In this chapter	547
	Overview	548
	Description of statistics	548

Header and trailer entries (Type 0)	560
Header entries (Type 0)	560
Recovery flags	561
Sequence numbers	561
Trailer entries (Type 0)	562
Recovery entries (Types 1–6)	562
System statistics entries (Type 8)	563
Final statistics	563
Partial statistics	563
Performance statistics	563
Entry formats	563
System final and partial statistics	564
System performance statistics	571
Entry format	571
Additional disk buffer monitor statistics	572
Multiprocessing (MP) subtask statistics	573
User statistics entries (Type 9)	574
User final and partial statistics	575
Final (logout) statistics	575
Partial statistics	576
User since-last statistics	579
Since-last statistics	579
Since-last high-water marks	580
Since-last in relation to final and partial statistics	581
Conflict statistics	584
User performance statistics	585
Parameters to activate performance statistics	585
Sampling	585
File statistics entries (Type 10)	586
Final statistics	586
Partial statistics	587
File statistics journal records layout	587
Text entries (Type 11, Type 13)	588
Noncontinued text entries (Type 11 — X'0B')	588
Possibly continued text entries (Type 13 — X'0D')	589
Initialization entries (Type 12 — X'0C')	590
Time-stamp entries (Type 14 — X'0E')	591
Merged journal bracketing entries (Type 15)	591
System Management Facilities	591
z/VM requirements	592
z/OS requirements	592
System Management Facility record layout and statistics	593
SMF logout record layout	593
SMF since-last record layout	597

B Coding SNA Communications Server Conversion Exit Routines

In this appendix	603
Overview	603
Rules governing data conversion exit routines	603
X3270CHK (check for terminal characteristics)	604
Communication protocols	605
X3270OUT (convert output from 3270 format)	606
X3270IN (convert input to 3270 format)	608
Communications protocols	608

C	ALLOCATE Utility in z/VSE	
	In this appendix	611
	Overview	611
	ALLOCATE control statement	611
	ALLOCATE z/VSE job stream.....	612

Index

About this Guide

Audience

The *Rocket Model 204 System Manager's Guide* is written for the system manager responsible for managing Model 204 system resources and coordinating user needs. The system manager should have system programming or database administration experience, including knowledge of job control (JCL or EXECs) and familiarity with standard IBM utilities.

Model 204 documentation set

The complete commercially released documentation for the latest version of Model 204 is available for download from the Rocket M204 customer portal.

To access the Rocket Model 204 documentation:

1. Navigate to:
<http://www.rocketsoftware.com/m204>
2. From the drop-down menu, select **Products > Model 204 > Documentation**.
3. Click the link to the current release and select the document you want from the list.
4. Click the .zip file containing the document.
5. Choose whether to open or save the document:
 - Select **Open** and double-click the pdf file to open the document.
 - Select **Save as** and select a location to save the zip file to.

Documentation conventions

This manual uses the following standard notation conventions in statement syntax and examples:

Convention	Description
TABLE	Uppercase represents a keyword that you must enter exactly as shown.
TABLE <i>tablename</i>	In text, italics are used for variables and for emphasis. In examples, italics denote a variable value that you must supply. In this example, you must supply a value for <i>tablename</i> .

Convention	Description
READ [SCREEN]	Square brackets ([]) enclose an optional argument or portion of an argument. In this case, specify READ or READ SCREEN.
UNIQUE PRIMARY KEY	A vertical bar () separates alternative options. In this example, specify either UNIQUE or PRIMARY KEY.
TRUST <u>NOTRUST</u>	Underlining indicates the default. In this example, NOTRUST is the default.
IS {NOT LIKE}	Braces ({ }) indicate that one of the enclosed alternatives is required. In this example, you must specify either IS NOT or IS LIKE.
item ...	An ellipsis (. . .) indicates that you can repeat the preceding item.
item ,...	An ellipsis preceded by a comma indicates that a comma is required to separate repeated items.
All other symbols	In syntax, all other symbols (such as parentheses) are literal syntactic elements and must appear as shown.
<i>nested-key</i> ::= <i>column_name</i>	A double colon followed by an equal sign indicates an equivalence. In this case, <i>nested-key</i> is equivalent to <i>column_name</i> .
Enter your account: sales11	In examples that include both system-supplied and user-entered text, or system prompts and user commands, boldface indicates what you enter. In this example, the system prompts for an account and the user enters sales11 .
File > Save As	A right angle bracket (>) identifies the sequence of actions that you perform to select a command from a pull-down menu. In this example, select the Save As command from the File menu.
E EDIT	Partial bolding indicates a usable abbreviation, such as E for EDIT in this example.

Part I

Model 204

Operational Issues

[Redacted]

[Redacted]

[Redacted]

This part describes the Model 204 runtime environment, how to define the environment for an end-user, managing system operations, managing Model 204 storage, groups, and files, server swapping, and application subsystems.

1

Model 204 Configurations and Operating Environments

In this chapter

- Overview
- About Model 204
- Model 204 configurations
- Operating environments
- 64-bit architecture support
- Model 204 storage

Overview

This chapter provides an overview of Model 204. The structure of the Model 204 database management system, basic Model 204 configurations, and the hardware and software environments required to run Model 204 are summarized.

Communication with Model 204 occurs by using Model 204 commands, parameters, files and file groups, procedures, and security features. The basic concepts for using these means of communication are given in the *Rocket Model 204 Parameter and Command Reference*.

About Model 204

Model 204 is a complete database management system that provides facilities for the creation, control, query, and maintenance of database files. Data intensive batch and online application systems can be developed with Model 204's self-contained User Language and embedded TP monitor. Application languages, such as Assembler, COBOL, PL/I, and FORTRAN can communicate with Model 204 functions through the Model 204 Host Language Interface. Model 204 supports SQL queries using the Horizon and CRAM interfaces.

Model 204 contains a system of files, tables, runtime parameters, user parameters, and software that frees users from the physical structure of the data and allows convenient, controlled access to that data.

- Sequential system and work files provide system input and output.
- Tables store information necessary for running specific requests.
- Runtime parameters handle execution time and system input stream specifications.
- User parameters control the environment of each online user.

A Model 204 database consists of files that utilize large fixed-length physical pages of 6184 bytes each. Each file is logically composed of four tables in which indexes and data are maintained. Entry order, unordered, sorted, and hash key file structures are supported.

Organization and maintenance of the data files are discussed in the *Rocket Model 204 File Manager's Guide*. Access and manipulation of data files are discussed in the *Rocket Model 204 User Language Manual* and the *Rocket Model 204 Host Language Interface Reference Manual*.

Staffing recommendations

Rocket Software recommends that each installation designate a person to be the Model 204 System Manager. The System Manager's responsibilities include:

- Installing and maintaining the Rocket Model 204 software
- Setting up and maintaining the OS or DOS JCL or CMS EXECs and Model 204 parameters for the Model 204 configurations used at an installation
- Setting up system security
- Acting as a liaison with Rocket Technical Support.

The System Manager's position is usually a part-time job in a small to medium size database environment. Often the System Manager also has the responsibilities of File Manager, also known as Model 204 database

administrator. If you are the System Manager only, Rocket Software strongly recommends that you study the *Rocket Model 204 File Manager's Guide* as well as the *Rocket Model 204 System Manager's Guide*, because you will be working closely with the File Manager.

Model 204 configurations

The basic Model 204 configurations are listed in Table 1-1. Various configurations, such as IFAM2 and ONLINE, can be combined to meet site requirements.

Table 1-1. Model 204 configurations

Configuration	Description
BATCH204	Handles a single user in batch mode.
ONLINE	Supports a batch user and a number of online users.
IFAM1	Supports host language calls to Model 204 files from multiple users. Programs are run as separate tasks in a single region, partition, or virtual machine.
IFAM2	Supports host language calls to Model 204 files from multiple users. Each program operates in its own region, partition, or virtual machine.
IFAM4	Supports host language calls to Model 204 files from multiple users. Programs are run as separate tasks in a single region, partition, or virtual machine.
BATCH2	Establishes a User Language connection to a Model 204 ONLINE running in a separate region.

Operating systems

Model 204 uses the same basic system architecture, file architecture, and User Language with each compatible operating system. Unique system characteristics that must be considered when running Model 204 are discussed, where appropriate, throughout this manual.

Model 204 is compatible with all versions of z/OS, z/VM, and z/VSE currently supported by IBM.

The Model 204 ONLINE configurations for each operating system are explained in detail in the appropriate *Rocket Model 204 Installation Guides*. Refer also to Chapters 2 through 4 of this manual for information on system and user commands and parameters.

Operating environments

Model 204 runs in an IBM mainframe environment with z/OS, z/VM, and z/VSE operating systems.

CPU

Model 204 runs on all mainframes that support z/Architecture principles of operations. Mainframes with ESA/390 architecture are supported from 9672 RA4 (Generation 3) and Multiprize 2000 and up.

MP/204 is an optional enhancement to Model 204 that provides full support for multiprocessor hardware configurations under z/OS.

31-bit addressing is also supported in all operating environments. When running in 31-bit mode, Model 204 allocates several data structures above the 16-megabyte line.

Model 204 supports 64-bit real storage and captured UCB allocated above the 16-megabyte line for z/OS and z/VM operating systems.

Direct Access Storage Devices (DASD)

Model 204 supports the devices listed in Table 1-2.

Table 1-2. DASD device types

Type	Model
CKD	3380, 3390
ECKD	9345
FBA	337x, 9332, 9335

Shared DASD and Global Resource Serialization

A shared DASD environment is one in which disk volumes may be shared between operating systems. Model 204 is supported under the three IBM operating systems: z/OS, z/VM, and z/VSE. A number of customers run Model 204, concurrently, under at least two of those three systems.

Many customers run Model 204 in the z/OS LPAR (Logical PARTition) environment where different versions of z/OS share disk volumes. In these systems, Model 204 jobs, ONLINE or BATCH, may be running concurrently in any of the defined LPARs and may be sharing disk volumes between those LPARs.

Model 204 file integrity in these multi operating system environments has always been provided through the Shared DASD Enqueue list, stored and maintained in the File Parameter List (FPL) page of each Model 204 file. Maintaining the Shared DASD Enqueue list, and therefore file integrity in a multi operating system environment, is totally dependent of the use of the device RESERVE macro.

The RESERVE macro provides exclusive access to a device. Therefore, any Model 204 job running under any operating system will have exclusive access to a Model 204 file during file open. File open is the event that adds, updates,

or removes entries in the SHARED DASD ENQUEUE list. These entries indicate which Model 204 job, virtual machine, or partition has the file open and whether that instance of Model 204 has share (read-only) or exclusive (update) access to the file. You can display these entries using the following command:

```
ENQCTL filename
```

It is therefore critical for Model 204 file integrity that the device RESERVE macro be allowed to function on all devices shared in a multi operating system environment where Model 204 jobs might run.

ENQCTL command enhanced

The ENQCTL command lists all entries in the shared DASD enqueue list that reside in a file's File Parameter List (FPL)—more than the previous maximum of eight entries.

Also, formerly when a file could not be opened due to shared DASD enqueueing conflicts, a maximum of eight entries was listed using the M204.0582 message number. With the enhancement to the ENQCTL command the M204.0582 message is issued as many times as necessary.

Cautions using Global Resource Serialization

The IBM Global Resource Serialization (GRS) feature available under z/OS may be configured to suppress the use of the device RESERVE macro. If this suppression is enabled, Model 204 file integrity cannot be guaranteed and data will become corrupted.

For this reason, it is imperative that the GRS Resource Name List (RNL) be configured to allow the RESERVE macro to be passed, in the channel program constructed by Model 204, to any device where a Model 204 file is allocated in a shared DASD environment.

For additional information regarding Model 204 files and shared DASD, please see:

- ENQCTL command, shared DASD enqueue control, in the *Rocket Model 204 Parameter and Command Reference*
- “Shared DASD locking conflicts” on page 36
- “CCATEMP in system recovery” on page 148

64-bit architecture support

Model 204 runs in 64-bit mode on z/OS and z/VM systems.

Understanding above the bar storage

In 64-bit mode z/OS and z/VM may create a region that has storage above the 2-gigabyte line called the bar. Address spaces with above the bar storage consist of three areas:

Storage location	Used for...
In 0-2 gigabyte range	Programs and data
Above 4 gigabytes	Data only
Note: in 2-4 gigabytes range	Unavailable for any purpose

Set the IBM MEMLIMIT system option

To implement above the bar storage IBM requires that you set a limit on how much of that virtual storage each address space can use. This limit is called MEMLIMIT. If you do not set MEMLIMIT, the system default is 0, meaning no address space can use above the bar virtual storage. To allocate Model 204 data structures such as buffer pool above the bar, MEMLIMIT should be properly set when running this release.

IBM provides several options to override the system default. Use one of the following options when you install and run Model 204 version 7.4.0:

- SMF MEMLIMIT parameter
- MEMLIMIT on JOB and EXEC statement
- MEMLIMIT environment through IEFUSI

z/OS customers running on z/OS 1.10 and later are afforded a default of 2 Gigabytes of above the bar storage per address space, replacing the previous default of 0 Gigabytes. Thus customers running on this operating system level do not need to set MEMLIMIT or REGION=0M to acquire above the bar storage.

Refer to IBM documentation about MEMLIMIT and limiting above the bar storage use in z/Architecture to implement the option that best meets your site requirements.

The use of real storage below the 2-gigabyte address is not controlled by MEMLIMIT. Only the amount of virtual storage is controlled by MEMLIMIT.

Model 204 entities in above the bar storage

The following Model 204 entities can be accessed directly in above the bar storage:

- Pages from Tables A, B, E, and X
- Index, procedure, and existence bit map pages from Table D

- CCATEMP pages with found sets, screens and images, and transaction backout log
- Saved compiled APSY procedures
- CCASERVER for swapped out users
- FTBL

Handling 64-bit statistics

To support very long running Model 204 regions, Rocket Software has modified the capacity of statistical counters by increasing the size of some statistics and also exploiting 64-bit processing where appropriate. For any in-house or third-party support applications that process statistical counters, you will need to review the statistics generated.

As some of the statistics fields are now double words, check *Appendix A: Using System Statistics* for the new layout of the System, Final and Partial statistics. Also, additional Disk Buffer Monitor, MP/204, and File statistics have been updated.

Look at your in-house or third-party support applications to see if you need to make changes because of the increased length of some of the statistics. Make any changes necessary to your applications, then reassemble with this new release.

Even if your in-house or third-party support applications do not refer to any of these double word statistics, you must reassemble your applications since all statistics offsets have changed.

Model 204 storage

The Model 204 program acquires all its working storage space dynamically. Working storage includes server areas, resource locking tables, buffer space, and control blocks.

The region or partition in which Model 204 runs must be large enough to contain the Model 204 code and the dynamically acquired working storage space.

Information about tracking dynamic storage allocation is provided in Chapter 18.

z/VSE storage considerations

The following considerations apply to a z/VSE environment:

- Model 204 allocates storage in the partition GETVIS area.

When running in z/VSE, the partition must be large enough to hold the executable phase (program) and have enough GETVIS area for Model 204 to allocate working storage.

- If the JCL EXEC statement SIZE parameter is missing, the size of the GETVIS area is equal to the default size (48K, unless defined in the IPL procedures).
- If there is insufficient storage, the SIZE parameter should specify partition space approximately equal to the program size, leaving a small (not less than 48K for z/VSE) partition GETVIS area.

In the case of an ONLINE system or an IFAM1 job, 48K of GETVIS area is not enough for an efficient system.

The SIZE parameter is strongly recommended for an ONLINE environment.

- The SIZE parameter is required when using IFAM1. If a PL/I Host Language program is used, Model 204 must take its dynamic storage from the partition GETVIS area through a SIZE parameter specification.

CMS disk considerations

Native CMS versions of Model 204 support files on variable-format disks and CMS-format disks. The disks can be partial- or full-volume minidisks, or dedicated volumes.

File location on the disks affects performance. Put all Model 204 files on variable-format disks for optimum access and performance.

- Use variable-format disks for files that experience high levels of activity. The Model 204 CMS interface supports asynchronous I/O operations through SIO-level logic and associated interrupt handling facilities. As a result, significant overlap between I/O and processing is achieved when variable-format disks are used.

Files on variable-format disks must be preallocated. A primary allocation is required. Secondary extents can be specified to permit limited extension of the file. File allocation information is recorded in the Volume Table Of Contents (VTOC).

- CMS-format disks affect performance by increasing I/O service time.
 - Additional arm motion and rotational delay caused by logical blocks of data spanning multiple physical blocks on a CMS-format disk extend the duration of I/O operations.
 - Synchronous DASD I/O operations (suspension of the virtual machine until the I/O operation completes) increase I/O service time.

Files on CMS-format disks do not require preallocation. Files increase dynamically as data is added, and are restricted in size only by the space available on the minidisk. New files are created automatically the first time they are referenced. Each CMS-formatted minidisk has its own Master File Directory (MFD) that contains allocation information for each file. When all the files are closed, the MFD maintained in virtual storage is transferred to disk. The CMS volume on which files exist is recorded in the MFD.

Related Rocket Software products

As the Model 204 system manager, you might be required to manage the following Rocket Software products, which can access Model 204 data.

Analytics/204

Analytics/204 is a PC-based visual data analysis tool that helps business users get to know their data, access their data, and trust their data, and without the need for IT assistance. Working in a Windows-based point-and-click environment, business users are able to:

- Acquire a better understanding of the data,
- Create subsets or segments of the data for targeted analytical or operational purposes, and
- Display or export the data in whatever format is required for maximum decision-making.

Connect★ and Horizon

Connect★ offers a client/server environment. On the client side, people use the PC software already familiar to them. On the server side is Model 204, ensuring high-performance database management. Connect★ unites the two through industry-standard SQL. Alternatively, using Rocket Software's innovative Remote Command Line facility, desktop applications can access Model 204 data through Rocket Software's User Language.

Horizon places Model 204 at the heart of a diverse client/server architecture. It connects all major IBM environments, as well as those from Sun, Digital Equipment, Tandem, Hewlett-Packard, and Data General, among others. With Horizon, you can leverage all your enterprise-wide computing resources in an integrated, distributed computing environment.

JDBC for Model 204

The JCBC for Model 204 product enables you to access Model 204 using the Java language. JDBC for Model 204 specifically supports Connect★.

MP/204

MP/204 enables a single copy of Model 204 to leverage multiprocessing configurations on IBM or compatible mainframes running z/OS/XA or z/OS/ESA. The result: significant performance enhancements in both system throughput and response times by utilizing parallel CPU processing.

MQ/204

MQ/204 provides User Language extensions to manipulate MQSeries message queues and receive and send data throughout your enterprise. Application developers are increasingly turning to Message Queuing Middleware (MQM) for program-to-program communication. MQM is especially useful for supporting applications that are distributed across different platforms and have only intermittent or long-distance network connections between them.

Parallel Query Option/204

With Parallel Query Option/204 (PQO) you can raise the performance of Model 204 by a factor of two, three, four or more. PQO lets you break up massive queries into smaller pieces, which are then all searched and sorted simultaneously in parallel. PQO finds answers faster because the requests access smaller sets of data. Requesters get their answers sooner and other users' queries don't wait as long in the queue.

You can share data between production, test, and even historical data regions. You can access Model 204 data residing on one region from any number of other Model 204 regions. Furthermore, PQO lets you distribute Model 204 regions within the same machine, on different machines, or even across different IBM operating systems.

2

Defining the Runtime Environment (CCAIN)

In this chapter

- Overview
- Structure of CCAIN
- ONLINE data streams with CCAIN
- CMS ONLINE command
- CMS utilities and EXECs
- BATCH204 JCL with CCAIN
- Runtime environment specifications
- Resource locking
- Disk buffers and Model 204 storage
- Managing delayed disk updates
- Understanding file statistics
- Server areas
- Server tables

Overview

CCAIN is the Model 204 control file used to define the runtime and user environments, and to control system operations.

When all the CCAIN input stream is read, Model 204 automatically terminates.

The discussion of the CCAIN file is presented in following chapters:

- This chapter describes the structure of the CCAIN file and the parameters used to define the runtime environment. Topics directly relating to the runtime parameters such as resource locking, disk buffers, and server areas are included.
- Chapter 3 discusses setting up the user environment.
- Chapter 4 discusses the commands used to control system operations.

Special considerations relating to Model 204 configurations or operating systems follow each discussion whenever appropriate.

Refer to the *Model 204 Parameter and Command Reference* for details about individual runtime parameters.

Structure of CCAIN

The CCAIN file is divided into three sections:

- Runtime environment specifications line, which is known as the **User 0 parameter line**, sets system characteristics and default user parameters during Model 204 initialization.

User 0, which acts as the system operator, is the name given to the input stream used by Model 204 initialization routines. The input stream is read from the CCAIN file and echoed on the CCAPRINT file (for more information on CCAPRINT, see Chapter 5).

The User 0 parameter line, which is described in this chapter, includes:

- System table sizes
- I/O buffer sizes
- Scheduler and performance options
- Recovery options

Parameters on the User 0 line are specified on the first line of the CCAIN input stream unless certain DEFINE commands are used. Commands such as DEFINE DATASET and DEFINE STREAM are the only User 0 statements that can precede the User 0 parameter line.

- User environment definitions and specifications are set for each user on a separate line for:
 - Device type and terminal communication network
 - Compiler table defaults

- Server configuration
- Default output options
- System control commands are entered on successive lines. System commands include:
 - Recovery procedures
 - Suspension of User 0
 - Message control
 - Allocation and freeing of data sets
 - Definition of data sets and printers
 - End-of-data and end-of-job statements
 - User Language requests
 - FLOD programs

ONLINE data streams with CCAIN

The following pages present z/OS, z/VSE, and z/VM examples for ONLINE data streams. The ONLINE examples illustrate the structure of CCAIN, including the runtime environment specifications line (User 0), user environment definitions and specifications, and system control commands.

A z/OS example for a BATCH204 run follows the ONLINE data streams.

Parameter lines

A CCAIN parameter line consists of one or more 80-column card images with parameter keywords and values in columns 1 through 71.

If the line exceeds 71 characters, any non-blank character in column 72 indicates continuation to the next card image.

The maximum length of the parameter area is controlled by the LIBUFF parameter, which is listed in Tables 2-4 and 2-6.

Model 204 accepts comment lines or blank lines after the User 0 parameters in the CCAIN input stream. IODEV lines can have comments and blank lines before, between and after them.

As shown in the following examples, these rules governing parameter lines apply in all operating environments.

z/OS JCL

The following example shows z/OS JCL for an ONLINE data stream containing the CCAIN file.

```
//M204ONLN JOB CLASS=A,MSGCLASS=A,NOTIFY=LEN
//RUN EXEC PGM=ONLINE,REGION=4096K,
```

ONLINE data streams with CCAIN

```
//          TIME=1440,PARM=('SYSOPT=149,LIBUFF=1024,LOBUFF=3000')
//STEPLIB   DD      DSN=M204.RLSE.LOADLIB,DISP=SHR
//CCAJRNL   DD      DSN=M204.JOURNAL,DISP=SHR
//CHKPOINT  DD      DSN=M204.CHKPOINT,DISP=SHR
//CCATEMP   DD      UNIT=3380,SPACE=(TRK,90),
//          DISP=(NEW,DELETE)
//CCASNAP    DD      SYSOUT=A
//SYSMDUMP   DD      SYSOUT=A
//SYSUDUMP   DD      SYSOUT=A
//CCAAUDIT   DD      SYSOUT=A
//CCASTAT    DD      DSN=M204.CCASTAT,DISP=SHR
//CCAGRP     DD      DSN=M204.CCAGRP,DISP=SHR
//CLIENTS    DD      DSN=M204.FILE.CLIENTS,DISP=SHR
//VEHICLES   DD      DSN=M204.FILE.VEHICLES,DISP=SHR
//CLAIMS89   DD      DSN=M204.FILE.CLAIMS89,DISP=SHR
//CLAIMS90   DD      DSN=M204.FILE.CLAIMS90,DISP=SHR
//CCASERVR   DD      UNIT=3380,DISP=(NEW,DELETE),
//          SPACE=(CYL,5,,CONTIG)
//CCAPRINT   DD      SYSOUT=A
//CCAIN DD *
USERS=9,NSERVS=2,MINBUF=18,MAXBUF=1000,
TERMBUF=5,NFILES=4,NDCBS=4,NDIR=4,SPCORE=50000,
IFAMBS=4000,LRETL=800,VTAMNAME=M204,CRFSCHNL=M204FULL,
CRIOCHNL=M204PROD,IFAMCHNL=IFAMPROD,
RCVOPT=9,CPMAX=1,CPTIME=30,CPTO=5,CPTQ=5,
LFSCB=7000,LGTBL=500,LSTBL=3000,LVTBL=200,
LOUTPB=3000,NBKPG=5,OUTCCC=80,SERVSIZE=72456
IODEV=7,NOTERM=3,POLLNO=1,SERVSIZE=72456
IODEV=7,POLLNO=2
IODEV=7,POLLNO=3
IODEV=29,NOTERM=2,POLLNO=1
IODEV=29,POLLNO=2
IODEV=11,NOTERM=2,POLLNO=1
IODEV=11,POLLNO=2
IODEV=23
HALT 27,MODEL 204 IS UP AND RUNNING,3,EOD
EOD
*SLEEP 300
BROADCAST URGENT 1***SYSTEM GOES DOWN IN 5 MINUTES
*SLEEP 300
HALT 24,WAIT FOR USERS TO LOGOUT,3,E0J
CHECKPOINT
BUMP ALL
*SLEEP 300
E0J
```

X Runtime
X environment
X definitions
X User0
X
X
X
User environment
defining 8 users
device types, and
the communication
network
System operation
control commands

z/VSE JCL

The following example shows z/VSE JCL for an ONLINE data stream containing the CCAIN File:

```
// JOB ONLINE
// DLBL M204LIB,'M204.PROD.LIBRARY'
// EXTENT SYSnnn,...
// LIBDEF PHASE.SEARCH=(M204LIB.V210,PRD1.BASE)
```

```
// DLBL CCAJRNL,'MODEL204.CCAJRNL'
// EXTENT SYS001,balance of extent information
// DLBL CHKPNT,'MODEL204.CHKPOINT'
// EXTENT SYS001,balance of extent information
// DLBL CCATEMP,'MODEL204.CCATEMP',,DA
// EXTENT SYS001
// DLBL CCASRV0,'MODEL204.CCASERV0',,DA
// EXTENT SYS001
// DLBL CCASRV0,'MODEL204.CCASERV0',,DA
// EXTENT SYS001
// DLBL CCAGRP,'MODEL204.CCAGRP',,DA
// EXTENT SYS001
// DLBL CCASYS,'MODEL204.CCASYS',,DA
// EXTENT SYS001
// DLBL CCASTAT,'MODEL204.CCASTAT'
// EXTENT SYS001,balance of extent information
// ASSGN SYS001,X'cuu'
// ASSGN SYS002,X'108' ---- Local 3270 terminal
// ASSGN SYS003,X'019' ---- Local 3270 terminal
// ASSGN SYS008,PRINTER ---- Audit trail
*** INSERT LABEL INFORMATION AND ASSIGNMENTS
*** FOR USER DATABASE FILES HERE
// UPSI 10011011
// OPTION SYSPARM='LIB=512'
// EXEC ONLINE,SIZE=AUTO
NUSERS=6,MAXBUF=1000,LFSCB=4900,           Runtime environment
LOUTPB=2400,NDIR=10,NSERVS=2,SERVSIZE=95000, definition (User 0)
RCVOPT=9
IODEV=35,INPUT=SYS002,SERVSIZE=95000      User environment
IODEV=35,INPUT=SYS003                     defining 5 users,
IODEV=41,NOTERM=2,POLLNO=1                device types, and the
IODEV=11,POLLNO=2                          communication network
IODEV=23,NOTERM=1,POLLNO=1
*** INSERT USER ZERO REQUESTS HERE
HALT 22,MODEL 204 IS AVAILABLE,3,EOJ      System operation
CHECKPOINT
EOJ                                         control commands
/*
/&
```

CMS CCAIN file

```
PAGESIZE=6184,NUSERS=5,NSERVS=2,          X
NFILES=3,NDCBS=3,MINBUF=18,MAXBUF=1000,   X
RCVOPT=9,SERVSIZE=95000
IODEV=41,NOTERM=2,POLLNO=1
IODEV=41,POLLNO=2
IODEV=39,NOTERM=1,POLLNO=1
```

```
IODEV=43,NOTERM=1,POLLNO=1
HALT 19,MODEL 204 IS NOW UP,10,END OF DAY
EOD
HALT 24,WAIT FOR USERS TO LOGOUT,3,EOJ
CHECKPOINT
EOJ
```

z/VM ONLINE processing

A Model 204 ONLINE environment is created in a z/VM environment by:

- Defining runtime parameters in the User 0 input file.
- Executing the “CMS ONLINE command” on page 18, which causes an EXEC procedure to:
 - Execute a user-created EXEC procedure to define the file recovery environment
 - Invoke Model 204 to perform file recovery
 - Execute a user-created EXEC procedure to define the ONLINE environment
 - Invoke Model 204 to establish the ONLINE environment

Samples of the components necessary to invoke a Model 204 ONLINE environment follow.

EXEC procedure defining the ONLINE environment

```
&TRACE OFF
&C FILEDEF * CLEAR
&C LABELDEF * CLEAR
&C FILEDEF CCAIN DISK DOCONLN CCAIN A
&C FILEDEF CCAPRINT DISK DOCONLN CCAPRINT A
&C FILEDEF CCAAUDIT J DSN M204 CCAAUDIT
&C FILEDEF CCASNAP PRINTER
&C FILEDEF CCAJRNL G DSN M204 CCAJRNL
&C FILEDEF CHKPOINT G DSN M204 CHPNT
&C FILEDEF CCATEMP H DSN M204 CCATEMP
&C FILEDEF CCASERVER I DSN M204 CCASVR
&C FILEDEF CCAGRP J DSN M204 CCAGRP
&C FILEDEF CCASYS J DSN M204 CCASYS
&C FILEDEF CCASTAT J DSN M204 CCASTAT
&C FILEDEF CARS J DSN M204 CARS
&STACK SYSOPT 146 LIBUF 1000
```

CMS ONLINE command

Purpose The ONLINE command brings up Model 204 in the service machine, allowing multiple users to log on. For example:

Syntax

The format of the ONLINE command is:

```
ONLINE [TEST] [NODCSS] [IFDIAL] [exec1] [BYPASS]
      [exec2] [BYPASS] [execargs]
```

Where

- *TEST* specifies that a TEST version of the Model 204 ONLINE module or shared segment, such as T204, is to be invoked. The default is the production version M204.
- *NODCSS* specifies that shared segments are not to be used, even though they exist.
- *IFDIAL* specifies that a single user IFDIAL connection is to be made (saved segments mandatory). The IFDIAL connection must be made on the main (nonrecovery) step.

A single-user IFDIAL EXEC procedure, SAMPDIAL, is supplied as part of the distributed material. Customize and install SAMPDIAL on an accessible minidisk. The M204 EXEC expects the IFDIAL EXEC to be named SINGDIAL.

For more information, refer to the *Model 204 z/VM Installation Guide*.

- *exec1* specifies the name of the EXEC procedure that contains the ACCESS commands for the required minidisks and the file definition (FILEDEF or M204FDEF) commands for Model 204 recovery purposes. You must create the EXEC in accordance with the file requirements for the Model 204 ONLINE environment to be recovered.
- *BYPASS* specifies not to use the EXEC procedure name in the ONLINE command and bypasses the recovery or Online steps, or both.
- *exec2* specifies the name of the EXEC procedure that contains the ACCESS commands for the required minidisks and the file definition (FILEDEF or M204FDEF) commands for Model 204 regular online production files. You must create the EXEC in accordance with the file requirements for the Model 204 online environment to be initiated.
- *execargs* are any user arguments, which are passed directly to the EXECs (1 and 2).

Usage notes

The EXEC can also include the Dictionary and Access/204 file definitions, if they are installed.

The following considerations apply to CMS online command processing:

- If no operands are specified on the ONLINE command, the default name of the restart EXEC procedure is M204REST.
- The default name of the Online EXEC procedure is M204DEF.
- If one operand is specified, it is assumed to be the name of the Online EXEC procedure.

- EXEC procedures invoked by ONLINE provide the necessary Model 204 parameters.
- Required options must be placed in the stack (the &STACK command) as keyword-value pairs, separated by blanks.
- If IFDIAL is specified, the main (nonrecovery) EXEC must provide only one parameter, the user program name, in the stack. The program name must be placed in the CMS stack before returning to the ONLINE EXEC.

The IFSETUP function (see the *Rocket Model 204 Host Language Interface Reference Manual*) is used to send the CCAIN parameters via the user program. Neither CCAIN nor CCAPRINT are used for IFDIAL connections.

- A single-user Model 204 interactive Online environment uses an EXEC procedure, SAMPSING, which is supplied as part of the distributed material. An IODEV statement is not required. (See Chapter 3.)

Customize and install SAMPSING and its companion, SAMPSING CCAIN, on a generally accessible CMS minidisk. Name the customized files SINGLUSR EXEC and SINGLUSR CCAIN to sustain compatibility with the standard distributed user interfaces.

Return codes are evaluated as follows:

- A return code of zero from any one of the EXEC procedures invokes Model 204.
- A return code of one (1) bypasses the invocation of Model 204.
- Any other return code is considered an error and causes the ONLINE EXEC to terminate immediately.

Example

```
ONLINE BYPASS DOCONLN
```

The parameters used in the sample above:

- Use the version of Model 204 with saved segments
- Provide no recovery arguments
- Execute a user-written EXEC (DOCONLN) that defines the ONLINE environment

CMS utilities and EXECs

Table 2-1 explains how to use CMS utility command modules relating to Model 204 in the CMS environment. None of the commands can be issued within the

Model 204 environment. For information on syntax, refer to the *Model 204 Parameter and Command Reference*.

Table 2-1. CMS utilities

Utility	Purpose	Comments
M204APND	Concatenates file definitions.	A DDNAME of CLEAR removes all file definitions.
M204CMS	The interface between CMS and Model 204 that provides system services during execution of load modules, for example M204ONLN, M204IFM1, and M204UTILC.	
M204FDEF	Creates file definitions for files on unaccessed variable format disks without accessing the resident disk.	An example defining the file CLI: CP LINK MVS 201 201 MW WRITE M204FDEF CLI 201 DSN M204 CLI
M204LDEF	Specifies magnetic tape label information for tape volumes using the M204APND module.	Standard LABELDEF command parameters and options are used.
M204UTIL	Initializes, labels, allocates, erases, renames, and lists variable-format volumes.	Not recommended for space allocation on DOS or OS format disks owned by a guest operating system, because the catalog is not updated and does not work on indexed VTOCs. You can use IEFBR14 for space allocation and M204FDEF or a FILEDEF for access.
M204XFER	Transfers control to the version of the M204CMS module that executes in a saved segment (DCSS).	M204XFER can also load a second DCSS containing the Model 204 program invoked by M204CMS.

Using the M204UTIL utility

The M204UTIL utility uses RESERVE/RELEASE logic when updating the VTOC of a variable-format disk. You can use it to manipulate a volume being used by one or more Model 204 service virtual machines. If you erase data sets that are in use by Model 204 on such volumes produces unpredictable results.

Table 2-2 lists the options that are available with M204UTIL.

Table 2-2. M204UTIL options

Allocation unit	Option
BLKSIZE	Block size
DSORG	Either PS or DA
LRECL	Record length
PRIMARY	CYL TRK BLK
RECFM	F/FA/FBA/V/VA/VBA/U

You must specify the primary option. Other options are not mandatory. The operands and options are function-dependent.

For example, the first statement below initializes a temporary minidisk. The second creates a data set named DEV.SCRATCH.CCATEMP to be stored on it. The third statement erases the data set.

```
M204UTIL INITIAL 291 TMP291
M204UTIL ALLOCATE DEV SCRATCH CCATEMP 291 (PRIMARY 5 CYL)
M204UTIL ERASE DEV SCRATCH CCATEMP 291
```

Using the M204MOUN EXEC to mount and dismount tapes

When a tape must be mounted, the CMS interface to Model 204 invokes the EXEC procedure M204MOUN, passing the DDname, device address, volume serial number, volume sequence, and access type (READ or WRITE) as arguments.

The M204MOUN EXEC determines the status of the tape device, issues appropriate Control Program and CMS commands, and sends a message to the system operator for a tape mount and drive attachment to the service virtual machine, if necessary.

Based on criteria at its disposal, the EXEC can reject the attempt to mount the tape. You can alter the M204MOUN EXEC defaults to meet site requirements.

When a tape volume is dismounted, the CMS interface to Model 204 invokes the EXEC procedure M204UNLD, passing the DDname, device address, volume serial number, volume sequence, access type (READ or WRITE), and file status (EOV or EOF) as arguments.

- EOV (request another volume) indicates an entry at end-of-volume.
- EOF (no further requests) indicates an entry due to end-of-file.

BATCH204 JCL with CCAIN

SAMPLE: z/OS JCL for invoking a BATCH204 run

```
//RUN      EXEC      PGM=BATCH204,REGION=1200K,
//          TIME=10,PARM='SYSOPT=144'

//STEPLIB   DD        DSN=M204.LOADLIB,DISP=SHR
//CCAAUDIT  DD        SYSOUT=A
//CCASTAT   DD        DSN=M204.CCASTAT,DISP=SHR
//CCAJRNL   DD        DSN=M204.JOURNAL,DISP=SHR
//CCATEMP   DD        UNIT=3330,SPACE=(TRK,20),
//          DISP=(NEW,DELETE)
//CCASNAP   DD        SYSOUT=A
//SYSMDUMP  DD        SYSOUT=A
//SYSUDUMP  DD        SYSOUT=A
//CENSUS    DD        DSN=M204.FILE.CENSUS,DISP=SHR
//CCAPRINT  DD        SYSOUT=A
//CCAIN     DD        *
                      PAGESZ=6184,NFILES=1,SNAPCTL=2

.
.
.
```

STEPLIB points to the load module library where the Model 204 program is linked.

- If the load module library is added to the LINKLIB concatenation, *STEPLIB* is not necessary.
- If EXCPVR under z/OS is used, *STEPLIB* must be authorized.

Runtime environment specifications

You can specify Model 204 runtime environment parameters on the EXEC statement and on the User 0 parameter line of the CCAIN input stream.

The following sections explain how to set EXEC parameters, discuss the most commonly used User 0 parameters, and detail procedures specific to z/VSE and z/VM environments.

For complete information on parameters, refer to the *Model 204 Parameter and Command Reference*.

Specifying EXEC statement parameters

The JCL EXEC statement includes the following parameters that affect the runtime environment (see the sample in the section “z/OS JCL” on page 15),

- *PGM* specifies the Model 204 configuration.
- *TIME* specifies how long Model 204 can run before being canceled by the operating system. Specify at least 10 seconds for system initialization and normal termination.

Under z/OS, if *TIME* is set to 1440, the operating system's automatic cancellation of the run is bypassed. If the Model 204 automatic shutdown facility is also bypassed, then Model 204 can run indefinitely until brought down by other means.

- *PARM* sets various Model 204 parameters, including the *LIBUFF* parameter and *SYSOPT* parameter, which is described in "SYSOPT parameter options" on page 25.

Note: The value set for *LIBUFF* takes effect immediately, so you must set *LIBUFF* large enough to accommodate *CCAIN*.

- *REGION* specifies the size of the memory area allocated for the Model 204 configuration. The next section explains how to estimate the value of this parameter.

Calculating region size

Consider the following factors when estimating region size for an Online system (z/OS) or z/VM machine size (CMS):

- Size of the load module, which varies depending on the use of optional modules
- Spare core (SPCORE) specification, the default is 8192 bytes

Increase the default, if you use deferred update, the FLOD exit (FLODXT) feature, directed output, or active subsystems. Additional memory is also required to open sequential data sets. The requirements for FLODXT are given under SPCORE in Table 2-4 on page 27.

- Number of buffers used for each server

Four buffers for each server is the minimum requirement. Each buffer requires slightly more than one Model 204 page. The main memory required is dependent upon the *SERVSZ* parameter setting.

No work space is required. Under normal conditions, five active users or application threads can be serviced efficiently by one server.

- Number of servers allocated

The size of each user's area is dependent on the settings of the compiler table parameters governing the size and complexity of the User Language requests. The formula to calculate server area size is given in the section "Sizing user server areas" on page 46.

In exceptional cases, processing needs might require a distinct server for each user.

- Storage overhead

Approximately 500 bytes per user and 700 bytes per file are required. The actual amount depends on the number of data sets and extents.

Setting the SYSOPT parameter

SYSOPT values, which can be summed, define actions taken during a run. Individual SYSOPT options are shown in Table 2-3.

Table 2-3. SYSOPT parameter options

Option	Specifies...
128	Log for the CCAJRNL and the CCAAUDIT data sets
64	Abend without a dump when the return code is nonzero
32	Print lines relating to system initialization or IFAM function calls (RK lines) on the audit trail or journal
16	Login is required
8	Automatic disconnect operation in response to the LOGOUT command
4	Execution of data definition commands within a particular run only through the File Management facility of Dictionary
2	Existing permanent group file (CCAGRP) is required
1	CCASYS file, which is required for subsystem applications

SYSIPT logical unit in z/VSE

In the z/VSE environment, CCAIN is replaced by the logical unit SYSIPT, a device-independent input reader. Typically, the SYSIPT logical unit is assigned to the same device as that used by the Job Control program for reading JCL. It is usually unnecessary to provide a z/VSE ASSGN statement for this logical unit.

You must place CCAIN data in one of the following:

- Job stream immediately following the z/VSE EXEC Job Control statement:

```
// EXEC ONLINE,SIZE=AUTO
  PAGESZ=6184,RCVOPT=1
  .
  .
  .
  .
/*
/ &
```

For a full example, see “z/VSE JCL” on page 16.

- A disk data set. Use any utility that takes card input and writes it to a file of 80-byte unblocked records. You must also supply a DLBL and EXTENT for either CCAIN or IJSYSIN:
 - For CCAIN, the symbolic unit referenced on the EXTENT statement must be a programmer logical unit, for example, SYS022.
 - For IJSYSIN, the symbolic unit referenced on the EXTENT statement must be SYSIPT.
- Procedure with the DATA=YES option on the CATALP statement.

User 0 input file in the z/VM environment

In z/VM, the User 0 output file is specified as a file stored on a minidisk. There are no restrictions on the choice of the CMS file identifier. For example:

```
FILEDEF CCAIN DISK DOCONLN CCAIN A
```

Model 204 also runs within a user's virtual machine in a single-user mode. In this case the file is defined with a FILEDEF command, similar to the example shown above, in the single-user EXEC. Runtime parameters are set up the same as for multiple users, except that the number of users is set to one (NUSERS=1).

Stacking z/VM runtime parameters

In a z/VM environment, the EXEC containing the FILEDEF commands can specify SYSOPT (and any other runtime parameters) before initialization by stacking the parameters:

```
&C FILEDEF CCAIN DISK DOCONLN CCAIN A
&C FILEDEF CCAPRINT DISK DOCONLN CCAPRINT A
.
.
.
*
*STACK THE PARM FIELD VALUE FOR MODEL 204
*
&STACK SYSOPT 128 LIBUFF 2048
```

User 0 parameters

The runtime environment specifications entered on the User 0 parameter line further define system options, user default values, and work area sizes.

Parameters common to many Model 204 configurations are summarized in Table 2-4. A full description of User 0 parameters can be found in the *Model 204 Parameter and Command Reference*.

Table 2-4. Common runtime parameters

Parameter	Specifies...
CFRLOOK	Collect critical file resources conflict statistics.
CPMAX	Maximum number of checkpoints
CPTIME	Checkpoint time intervals
LENQTBL	Number of entries in each user's resource enqueueing table. (See page 2-22 for sizing formulas.) The default is 6.
LIBUFF	Length of the input buffer used for input lines from CCAIN or the user's terminal. LIBUFF must be three bytes longer than the longest line or record read into it. Longer lines are rejected with an error message. The default is 255. If an input line is continued with a nonblank character, the number of characters in the original line and all continuations (not including continuation characters) must fit in the LIBUFF specification.
LOBUFF	Length of the output buffer used for output lines to CCAAUDIT, CCAPRINT, for a user's terminal, or for a directed output (USE) data set. LOBUFF can be reset on individual user parameter lines. The default is 256. The recommended value for SQL processing is 5000.
LOGADD	Number of slots reserved for adding new password (CCASTAT) entries. The default is 0.
LOGFAIL	Action taken when the number of consecutive login failures exceeds the value of the LOGTRY parameter. Default is 0.
LOGONENQ	Use of unique user IDs for systemwide logons to a single ONLINE system. (See the LOGONENQ entry in Table 3-11 on page 105 to specify unique user IDs for specific terminals.) The default is 0. The Subsystem Management facility is not affected by LOGONENQ.
LOGTRY	Maximum number of login attempts allowed. The default is 0.
LRETBL	Length of each user's part of the record enqueueing table. (See "Resource locking" on page 31.) Default is 200.
LRUTIM	Disk page reference interval for references considered obsolete. Use LRUTIM to calculate DKRR statistics. The default is 0.
MAXBUF	Maximum number of file page buffers allocated during Model 204 initialization. (See "Disk buffers and Model 204 storage" on page 37.) The default is 256.
MINBUF	Minimum number of file page buffers allocated during Model 204 initialization. (See "Disk buffers and Model 204 storage" on page 37.) The default is 18.

Table 2-4. Common runtime parameters (Continued)

Parameter	Specifies...
NDCBS	Number of Model 204 file DCBs that can be used at any one time. The default is 10.
NDIR	Maximum number of Model 204 files that can be opened during a run. The default is 5.
NFILES	Maximum number of Model 204 files that can be open at any one time. Files remain open until an explicit CLOSE is issued or the session ends. The default is 2. NFILES, NDCBS, and NDIR specifications are automatically incremented during system initialization if SYSOPT=2 (permanent group file). With the VIEW command, you can view parameter values during a run.
NGROUP	Maximum number of file groups each user can have open at the same time. The default is 5.
NSERVS	Number of servers. The default is NUSERS (see below).
NSUBTKS	Maximum number of pseudo subtasks that can be generated in a Model 204 run. The default is 4.
NUSERS	Total number of User Language users and IFAM2 or IFAM4 threads supported. The default is 1. The value of NUSERS consists of the total number of I/O device types (IODEVs) specified on the user parameter lines plus 1 for User 0. Online users are defined as terminal users with a particular type of terminal or as a host language thread, if IFAM is supported in the online region.
PAGEFIX	Fixes areas in memory; can be useful in reducing paging traffic under z/OS or z/VSE. The default is X '0000000', which means nothing is page fixed. For z/VSE only: A PAGEFIX request is valid only if real pages have been assigned by the ALLOC command. If ALLOC is 0 or if the total number of pages to be fixed exceeds the number of real pages, the PAGEFIX request fails.
RETRVKEY	PF key used to retrieve a previous terminal input line. 280 bytes of spare core is required for each user that has a defined retrieve key. The default is 0.
SEQOPT	Activation of the prefetch (look-ahead) feature for User Language applications requiring entry order retrieval. Values are 1 (on) or 0 (off). The default is 0. Activation of SEQOPT requires resizing the MAXBUF parameter by using the formula: $\text{MAXBUF} = \text{NUSERS} * (4 + 2 * [\text{maximum FOR EACH RECORD loop nest level}])$ You can modify SEQOPT with the RESET command.

Table 2-4. Common runtime parameters (Continued)

Parameter	Specifies...
SERVSZ	Size of each server area. The default is 0. (See "Server areas" on page 46.)
SPCORE	<p>Minimum amount of storage within the Model 204 address space to leave unallocated at the end of Model 204 initialization. You can set SPCORE in the EXEC statement PARM field or on the User 0 parameter line. The default is 8192.</p> <p>Spare core is used by:</p> <ul style="list-style-type: none"> • IFAM4 application program storage. The default is 12288. • Active subsystems. (See the formula given in "SPCORE size" on page 191.) • FILELOAD or FLOD commands for tape input and deferred update output. When using the FLODXT feature, you must allocate 100 bytes for each FLODXT program. • Each defined retrieval key for previous terminal input (180 bytes per key). <p>In z/OS, a number of bytes of virtual storage equal to your SPCORE setting is reserved above the line, and the same number is reserved below the line.</p>
TIMESTOP	<p>Amount of time (CPU milliseconds) before automatic termination processing begins or before initiation of commands or User Language requests stops. TIMESTOP cannot be reset. The default is 1500.</p> <p>If TIMESTOP is set to 0, the Model 204 timing facility's automatic shutdown is bypassed. If, in addition, the JCL EXEC statement parameter TIME is set to 1440, the z/OS automatic shutdown is bypassed, and Model 204 continues to run indefinitely until brought down by other means.</p>
XMEMOPT	<p>Model 204 Cross-Memory Services Facility used by Timer PC and IOS BRANCH ENTRY for z/OS systems, as well as using SUSPEND/RESUME instead of WAIT/POST for communication between Model 204 real subtasks. Coordinates with XMEMSVC.</p> <p>It is also required for:</p> <ul style="list-style-type: none"> • IOS Branch Entry • UL/DB2 • CRAM-XDM • CPUID check <p>To choose the correct setting for your site, see the <i>Rocket Model 204 Parameter and Command Reference</i>, "XMEMOPT: Cross-memory services options".</p> <p>VIO is incompatible with IOSB and EXCPVR.</p>

Table 2-4. Common runtime parameters (Continued)

Parameter	Specifies...
XMEMSVC	<p>SVC number of the Model 204 Cross-Memory Services. Coordinates with XMEMOPT. The default is 0. It is also required for:</p> <ul style="list-style-type: none"> • IOS Branch Entry • UL/DB2 • CRAM V4.1 feature • CRAM-XDM • CPUID check

z/VSE UPSI Job Control statement

In a z/VSE environment:

- Use the SYSPARM parameter of the z/VSE OPTION Job Control statement to specify a limited number of parameters that are normally specified on User 0 or any user's parameter lines, if the maximum length of the data does not exceed eight characters. For example, you can set the LIBUFF parameter with the following statement:

```
// OPTION SYSPARM='LIB=2048'
```

- Set the SYSOPT value before initialization by using the z/VSE UPSI Job Control statement. Specify the value as a series of weighted UPSI bit values.

For example, including RK lines on the audit trail can be specified with the following statement, which is the equivalent of SYSOPT=32:

```
00100000
```

Table 2-5 summarizes the UPSI bit settings and SYSOPT equivalents.

Table 2-5. UPSI/SYSOPT settings

UPSI setting	SYSOPT value	Meaning
10000000	128	Generate audit trail and/or journal
01000000	64	Abend without a dump when the return code is nonzero
00100000	32	Include RK lines on the audit trail
00010000	16	Login required
00001000	8	Automatic disconnect in response to the LOGOUT command
00000100	4	Restrict the use of data definition commands within a run

Table 2-5. UPSI/SYSOPT settings

UPSI setting	SYSOPT value	Meaning
00000010	2	Open CCAGRP for use of permanent file group
00000001	1	Open CCASYS for use of application subsystem definitions

Resource locking

To maintain data integrity, resource locking requests and reserves system and file resources for share (SHR) and exclusive (EXCL) use with other users. Data corruption is prevented by using linked lists of system and file resources. Conflict in the locking table results from attempts to lock exclusively on a file resource.

Resource locking messages indicate a wait for a file, a conflict for a resource, or that the table is full and needs to be increased with the LENQTBL parameter.

Locking occurs on:

- File resources, which are usually locked in SHR mode, such as:
 - File access
 - Record locking table
 - Table B and Group index
 - Tables C and D
 - Permanent procedures
 - Access Control Table
- System resources, which are usually locked in EXCL mode, such as:
 - Access to the CCASTAT file
 - Group definition table in CCATEMP
 - Updates to CCAGRP
 - Names of a file and subsystem
 - User defined resources
- Records

The following sections explain the resource locking tables and the details of resource locking on single and multiple CPUs.

Record locking table

The record locking table, whose length is the product of the number of users (NUSERS) and the length in bytes of one user's part of the table (LRETBL), contains control information necessary to detect conflicts between users trying to update records simultaneously.

You can issue a MONITOR ENQ command to determine current usage in the record locking table.

Calculating allocated size of record locking table

The amount of space required by a request is roughly proportional to the number of lists and FIND statements contained in the request. Each FIND statement or list requires about 46 bytes per file for files less than or equal to 300,000 records. Space requirements increase at the rate of 2.25 bytes per segment. The maximum value is 65,535.

SYSOPT2=X'40'

Record sets—found sets, including FDWOL found sets, sorted sets, lists, and LPU lists—are traced through entries in the record locking table. One entry is required for each segment (49,152 records) in the record set. These entries are CCATEMP page numbers.

When SYSOPT2=X'40', the entries contain 4-byte CCATEMP page numbers. Setting SYSOPT=X'40' provides a substantial increase in the number of simultaneous record sets that can be concurrently active in a given Model 204 run. Therefore, if you set SYSOPT2=X'40', you should also at least double LRETBL.

- When the SYSOPT2 setting *does not* include X'40', then at any given time the bit maps corresponding to all users holding found sets of any kind must fit into CCATEMP pages designated as the small model page pool no matter how large CCATEMP has been allocated.
- When the SYSOPT2 setting *does* include X'40', the CCATEMP page restriction is removed and user found sets can be placed anywhere within CCATEMP. This includes both the small model page pool and the CCATEMP expansion area, allowing for the possibility of a greater number of concurrent found sets being held by all users.

Resource locking table

You can use the \$CENQCT function to obtain information on the number of unused entries in the resource locking table.

Sizing the resource locking table

For details on estimating the size of the resource locking table, see the description of LENQTBL in the *Rocket Model 204 Parameter and Command Reference*.

For z/OS, Model 204 allocates the resource locking table above the 16-megabyte line.

Multiple jobs running on one CPU

Locking occurs at the file level when application files are shared between multiple jobs.

File locking modes is EXCL when:

- Files are opened from a User Language thread or an IFAM1 thread that has file update privileges.
- Files are opened from an IFAM2 thread or an IFAM4 thread that has allowed updates with thread update and file update privileges.
 - Command is entered to create or delete a permanent group.

File locking is SHR mode when:

- Files are opened under any other circumstances than those listed above.
- Files are opened in deferred update mode. Such files remain in SHR mode for the duration of the run.

Note: Up to 192 files can be opened in deferred update mode. An attempt to exceed 192 files results in an error message.

- Files have the file recovery option (FRCVOPT) parameter set to include X '10'. Locking on an application file does not occur when it is closed, unless FRCVOPT is set to X '10'. (FRCVOPT is discussed fully in the *Model 204 File Manager's Guide*.)
- Model 204 job uses the CCAGRP data set.

The following sections explain how locking conflicts are handled by Model 204, and how data integrity is ensured when multiple jobs run on one CPU.

Handling locking conflicts

Locking conflicts for application files are handled first by the operating system and next by Model 204.

The operating system initially examines the disposition for all application files, as specified in the JCL for a job. If two jobs specify SHR for the same application file, the operating system allows both jobs access to the file. When

the second job attempts to process the application file, Model 204 determines that another job poses a locking conflict.

Model 204 reads the first page of the file and examines the lock, which is located on that page. If a conflict is detected, Model 204 waits until the job's time limit is reached for the file to become available. Then, if the file is not available, Model 204 sends error messages to the operator's console and to the output device of the user who attempted to open the file.

Error messages are issued once every five minutes until the file becomes available, the job time limit for an online job is reached, or the maximum number of error messages for a batch job (ERMX) is reached.

Messages sent to the operator's console are:

- For an Online job:

```
*** M204.0582: ACCESS TO FILE filename PREVENTED BY:
jobname
*** M204.0584: FILE IS IN USE -- filename
```

- For a batch job the message sent from Model 204 to the operator:

```
*** M204.0582: ACCESS TO FILE filename PREVENTED BY:
jobname
*** M204.0581: ENQ'ING TO SHR FOR FILE filename volname

*** M204.0582: ACCESS TO FILE filename PREVENTED BY:
jobname
*** M204.0583: ENQ'ING TO EXCL FOR FILE filename vol-
name
```

Data integrity

When multiple jobs run on the same CPU, data integrity is ensured by using:

- Operating system locking and unlocking facility for shared application files and the system file containing file group definitions (CCAGRP).
- Special lock stored in the system file containing user and file security information (CCASTAT).
- Restriction on sharing the system file that provides space for user work tables (CCASERVER) and the system scratch file (CCATEMP) between jobs.

Multiple Model 204 versions running on separate CPUs

The operating system file locking mechanism prevents concurrent updating and retrieval of data sets by jobs that run on separate CPUs. The RESERVE/RELEASE hardware feature restricts use of a device to a single CPU in the following ways:

- Device containing a file's first data set can be reserved when control of the file is gained on one CPU and one of the following conditions is true:
 - File is opened in a Model 204 job for the first time.
 - File that was read-only is first opened for update.
 - Last updating user closes the file.
 - File is completely closed.
- Device is released in each case after a single disk read and disk write have been performed.

Each file contains a list of jobs that have control of the file. The list is read and updated only while the device is reserved. If control of the file cannot be obtained, then the list is not updated, and the list of jobs preventing access, with their system IDs, is sent to the operator.

Each list entry contains the following information:

- SMF system ID lock type (SHR or EXCL)
- Job and step names
- Date and time that the list entry was created

How Model 204 resolves locking conflicts

Locking conflicts can be handled automatically or by issuing the ENQCTL command (see *Model 204 Parameter and Command Reference*).

Conflicts are handled automatically in single CPU error cases where a Model 204 job has a file open and either the operating system crashes or the Model 204 job is canceled. In these instances, the locking list in the file still shows the file as locked and the following process occurs:

1. When a file is opened, the locking list is processed after the operating system enqueueing. If the operating system enqueueing succeeds, there is no conflicting job on the requesting job's system. Any conflicting list entries for the same system are obsolete.
2. If the locking request is exclusive, any list entries for the current system are eliminated as obsolete.
3. If the request is shared, any exclusive entries for the current system and any shared entry having the same system ID, job, and step names are eliminated.

Locking conflicts between CPUs

You can clear conflicts occurring between CPUs by issuing the ENQCTL command to interrogate or modify the status of a file's locking list. The following rules apply to the ENQCTL command:

- If an ENQCTL command is issued with only a file name, all list entries for the file are displayed.
- If an ENQCTL command is issued with arguments, all the entries satisfying the arguments are deleted.

For example, if a system crash occurs for system S133, the operator at another system can issue the following command to remove all the locking list entries from CENSUS that were added by jobs running on system S133:

```
ENQCTL CENSUS S133
```

Indiscriminate use of the ENQCTL command can cause shared DASD integrity exposure through the removal of entries of active systems or jobs.

Shared DASD locking conflicts

If a shared DASD locking conflict occurs, Model 204 sends an error message to the operator's console. Error messages are issued once every five minutes until the conflict is resolved and the file becomes available.

Messages sent to the operator's console are:

- For a batch job:

```
*** M204.0582: ACCESS TO FILE filename PREVENTED BY:
jobname
*** M204.0584: FILE IS IN USE -- filename
```

Note: Both of these messages increment the error count for the batch job. Because the batch OPEN command is aborted after the specified maximum number of errors (ERMx) is reached, Model 204 does not wait indefinitely for the conflict to be resolved.

- For active system or job locking list entries deleted by using the ENQCTL command:

```
*** M204.0585: SHARED DASD ENQ LIST OVERLAID FOR
filename AT hh:mm:ss ON yy.ddd
```

The date and time identify the most recent update of the file. Used in conjunction with the operating system job logs, this information can determine the cause of the problem.

z/VSE considerations

The following considerations apply to resource locking in a z/VSE environment:

- File locking is available for z/VSE releases that support LOCK and UNLOCK (SVC 110).
- Multiple copies of Model 204 running in separate z/VSE systems cannot share any Model 204 database files.

Resource locking in z/VM

The following considerations apply to resource locking in z/VM environments:

- CMS-format disks cannot be shared in read/write mode by multiple virtual or real machines. Any attempt at SHR access destroys the data.

The file allocation techniques that are used and the lack of support in CMS for access serialization prevent effective read/write file sharing.

Files on CMS-format disks do not require preallocation. The files are created automatically the first time they are referenced and continue to increase in size as more data is added. File size is restricted only by the available space defined on the minidisk.

- Several virtual machines can share variable-format disks by using virtual RESERVE/RELEASE facilities.

RESERVE/RELEASE permits access to a volume restricted to a particular (real or virtual) access path. Because allocations are static in nature, a file can be read and written without further reference to the allocation information, unless secondary allocation functions are required.

Files on variable-format disks require preallocation. A primary allocation must be provided. Secondary extents can be specified to permit limited extension of the file. The file allocation information is recorded on a disk in the Volume Table of Contents (VTOC).

Some files on variable-format disks can be read/write shared by multiple Model 204 virtual machines while others cannot:

- Files that can be shared are CCAGRP, CCAIN, CCASTAT, CCASYS, and Model 204 files.
 - Files that cannot be shared are CCAAUDIT, CCAJRNL, CCAPRINT, CCARF, CCASERVER, CCASNAP, CCATEMP, CHKPOINT, RESTART, USE data sets, and deferred update data sets.
- SHR mode access on a read-only device can cause data inconsistencies.
If a CMS user has SHR access to a Model 204 file on a read-only minidisk, SHR does not prevent another user from upgrading to the EXCL mode.

Disk buffers and Model 204 storage

The disk buffer pool holds pages from database files and from CCATEMP and CCAGRP. CCATEMP pages consist of found sets, sorted sets, backpage images, temporary procedures and other data structures. Database pages consist of pages from Tables FCT, A, B, C, D, E and X. Pages are read into and written from this buffer pool by the disk buffer manager which manages this resource using a least recently used algorithm. This pool of buffers is shared by all users.

Model 204 utilizes the following areas of storage, depending on the operating system architecture your site supports:

- Below the line, for 24-bit storage for non XA systems: VSE, VM, and OS
- Above the line, for sites that support 31-bit storage for z/OS, OS/390, XA, ESA, z/VSE, z/VM
- Above the bar, for sites that support 64-bit storage for z/OS and z/VM

The buffer pool consists of the following data structures:

- Disk Buffer Control Blocks: 160 bytes each, one per buffer
- Hash cells: 16 bytes each. The number of hash cells allocated for each buffer is equal to the HASHCELL parameter which defaults to 3. For more details about the HASHCELL parameter see the *Rocket Model 204 Parameter and Command Reference*.
- Page fix lists - 16 bytes per buffer
- Buffers - 6184 bytes, plus 8-byte overrun detection area

The disk buffer overrun detection area, the space between each buffer in the disk buffer pool, is eight bytes of hexadecimal FF, so for each buffer 6192 bytes is allocated. The total size of the buffer pool is then:

$$(\text{NUMBUF} + \text{NUMBUFG}) * (6192 + 160 + (16 * \text{HASHCELL}) + 16)$$

NUMBUF is a viewable parameter and is equal to the number of buffers allocated above the line. NUMBUF may be equal to or less than MAXBUF, depending on the amount of virtual storage available to the job.

NUMBUFG is a settable and viewable parameter, equal to the number of buffers allocated above the bar. If NUMBUFG buffers cannot be allocated in available above the bar storage, then the run terminates.

Buffers in an ONLINE configuration

The following considerations apply to the use of buffers above the line in an ONLINE configuration:

- A minimum number of buffers is required for the run to come up. MINBUF, if set smaller than the result of the following calculation, is reset to this value:

$$\text{NLRUQ} * ((\text{NSERVS} + \text{NSUBTKS}) * \text{MAXOBUF} + 15)$$

- MAXBUF, if smaller than MINBUF after the previous calculation, is reset to the value of MINBUF.

Rocket Software recommends that you start with a minimum setting of MAXBUF=10000 and monitor performance statistics to determine if that number is adequate. Generally, performance will improve as the size of the buffer pool increases. That will not be the case, however, if real storage is limited and system paging increases. Many sites are running with MAXBUF=50000, 100000 and more.

Using 31-bit storage

In systems that support 31-bit addressing, Model 204 disk buffers are allocated above the 16-megabyte line. This frees virtual storage for other data that must remain below the line and allows for the allocation of a larger buffer pool since there is more virtual storage above the line. As the number of buffers increases, database pages can remain in memory for longer periods of time and repeated reads (I/O) for the same pages are reduced.

- If IOS BRANCH ENTRY (XMEMOPT=2) is used, control blocks, hash cells, and page fix lists are allocated above the line.
- If IOS BRANCH ENTRY is not used, the disk buffer control blocks and hash cells (and page fix lists if EXCPVR is used) are allocated below the line. The buffers themselves are allocated above the line.

HASHCELL - Allocation of hash cells per buffer pool page

You can control the number of hash cells allocated in the hash table with the HASCELL parameter. The hash table is used to locate pages in the buffer pool based on the file and page number. The default, and minimum, is three hash cells per page. In this release you can allocate as many as seven hash cells per page.

Using a higher value will:

- Reduce the number of hash cell collisions and thus reduce the CPU consumed to resolve any collisions.
- Result in the use of more virtual storage for the increased number of hash cells.

If NUMBUFG is also set to a value greater than zero, to allocate buffers above the bar, the hash cells are also allocated above the bar, saving below the bar storage. The total amount of storage required for hash cells can be calculated using the following formula:

$$16 * HASHCELL * (NUMBUF + NUMBUFG)$$

Using 64-bit storage

In systems that support 64-bit virtual storage, you can place Table B and Table X pages (and other entities listed in “Model 204 entities in above the bar storage” on page 8) in the above the bar buffer pool, or above the two gigabyte (2GB) address line. Pages that are not stored above the bar reside in the buffer pool above the line.

64-bit features are not yet available on IBM z/VSE systems.

- In most cases, Table B pages constitute the biggest portion of all pages in the buffer pool. Moving Table B pages to an above the bar buffer pool lets Model 204 place more pages from all other tables in the below the bar

buffer pool and thereby reduce I/O and CPU time to read and write pages to and from disk.

- When a buffer is allocated above the bar, the corresponding disk buffer control blocks (one per buffer, 160 bytes each) and hash cells (three per buffer, 16 bytes each) are also allocated above the bar. This means there is no below the bar storage penalty for allocating above the bar buffers.
- Having these two buffer pools rather than one improves Model 204 scalability by reducing MP collisions when using buffer pool resources.
- Eight bytes have been added to the end of every buffer, above and below the bar, to detect buffer overruns. The new buffer size per page is 6192 bytes (or 6184 plus 8).

Managing above the bar storage

When NUMBUFG is set to a nonzero value, an above the bar buffer pool is allocated with NUMBUFG buffers. This is in addition to the below the bar buffer pool which is always allocated with at least the minimum number of buffers, calculated as follows:

$$NLRUQ * ((NSERVS + NSUBTKS) * MAXOBUF + 15)$$

Table B and Table X pages, (and other entities listed in “Model 204 entities in above the bar storage” on page 8) use the above the bar buffer pool. Those pages are not read into the below the bar buffer pool. Consequently, most sites can reduce the size of the below the bar buffer pool by the high water mark of Table B pages currently resident in that buffer pool.

To quickly implement the above the bar feature initially set NUMBUFG equal to your MAXBUF setting and leave MAXBUF at its current setting.

The minimum number of above the bar buffers calculated by Model 204 uses the following formula:

$$NLRUQ * ((NSERVS + NSUBTKS) * MAXOBUF + 15)$$

If you set NUMBUFG to a lower value, it is reset to the calculated value.

If NUMBUFG is greater than zero, the buffer hash pool is allocated above the bar. In addition, control blocks associated with above the bar buffers are also allocated above the bar. NUMBUFG is limited to buffer pools of 4.2 terabytes or fewer.

To use above the bar buffer pool in z/OS, IOS Branch Entry is required. This means XMEMOPT must be set to include X'02'. You can explicitly exclude allocating above the bar buffers by setting NUMBUFG=0.

If NUMBUFG is greater than zero and XMEMOPT does not include X'02', the following message is issued, NUMBUFG is not reset, and the job terminates.

```
M204.2581: XMEMOPT=2 (IOS BRANCH) REQUIRED FOR NUMBUFG > 0
```

If you cannot get the number of buffers you requested, the job fails.

Determining NUMBUFG setting

The number of buffers you want to allocate above the bar and below the bar is dependent on the mix of work that is being done on your system. See “Model 204 entities in above the bar storage” on page 8 for a list of entities that can go above the bar.

- The LDKBMWNG parameter, which applies to above the bar buffers, corresponds to the LDKBMWND parameter, which applies to below the bar buffers.
- If NLRUQG is set greater than 1, then the value of LDKBMWNG is rounded up to a multiple of NLRUQ. LDKBMWND has a minimum size of one (1).

High values of LDKBMWNG might unnecessarily increase the number of writes done (measured by the DKWR statistic). Low values might cause excessive waiting for buffers (measured by the MAXIOX statistic). Rocket Software recommends starting values for LDKBMWND and LDKBMWNG at 10% of NUMBUF and NUMBUFG, respectively.

If you do not set LDKBMWNG, it is set to the same value as LDKBMWND.

Above the bar storage for EBM pages

Existence Bit Map (EBM) pages reside in above the bar storage.

Each Model 204 file contains one EBM page for each segment in a file. If a file has five segments, that means there are five EBM pages for that file.

- The NUMBUFG parameter specifies the number of buffers allocated above the bar. You can increase your NUMBUFG setting to allow more above the bar buffers for the EBM pages. Increase the allocation of NUMBUFG by a value that accommodates all the EBM pages for all files that might be open concurrently in your job.
- The MAXBUF parameter specifies the maximum number of buffers to be allocated below the bar. The NUMBUF parameter (view only) indicates how many buffers were actually allocated. You can reduce MAXBUF by the same value you used to increase NUMBUFG for EBM pages.

Above the bar storage for procedure pages

Procedure text pages, located in Table D, are also eligible to reside in the above the bar buffer pool. Each User Language procedure is stored in one or more procedure text pages, the initial page being pointed to via the procedure dictionary. (Pages from the procedure dictionary, which is also stored in Table D, are read into the below the bar buffer pool.)

This change is more likely to affect development environments than production environments, but in those development environments where NUMBUFG is

allocated, you might still want to tune NUMBUFG up and NUMBUF down accordingly.

Screens and images above the bar

Pages used for Model 204 SCREEN and IMAGE items now reside in the buffer pool above the bar.

Allocating the above the bar buffer pool

If NUMBUFG is set to a nonzero value, an above the bar buffer pool is allocated with NUMBUFG buffers. This is in addition to the below the bar buffer pool which is always allocated using the MINBUF and MAXBUF parameters previously discussed.

When the above the bar buffer pool is allocated, Table B and Table X pages use it exclusively. Those pages are no longer read into the above the line buffer pool. Consequently, you can reduce the size of the above the line buffer pool by the high water mark of Table B pages previously resident in that buffer pool.

IOS Branch Entry is required when NUMBUFG is greater than zero, so the XMEMOPT setting must include the x'02' bit.

In addition, the disk buffer control blocks associated with the above the bar buffers are also allocated above the bar.

Implementing above the bar use of storage - NUMBUFG

If you set NUMBUFG greater than zero to use storage above the bar, IBM requires that you set a limit on how much of that virtual storage each address space can use. This limit is called MEMLIMIT. See "Set the IBM MEMLIMIT system option" on page 8 for more information.

HASHCELL - Allocation of hash cells per buffer pool page

You can control the number of hash cells allocated in the hash table with the parameter HASHCELL. The hash table is used to locate pages in the buffer pool based on the file and page number. The default, and minimum, is three hash cells per page. You can allocate as many as seven hash cells per page. Rocket Software recommends resetting the default only when running with AMPSUBS>0.

Using a higher value will:

- Reduce the number of hash cell collisions and thus, reduce the CPU consumed to resolve any collisions.

Note: the DKSRHC statistic is not longer collected.

- Result in the use of more virtual storage for the increased number of hash cells. Each hash cell is 16 bytes and HASHCELL number of hash cells are allocated per buffer. The default value of HASHCELL is 3.

If NUMBUFG is also set to a value greater than zero to allocate buffers above the bar, the hash cells are also allocated above the bar, saving below the bar storage. The total amount of storage required for hash cells can be calculated using the following formula:

$$16 * \text{HASHCELL} * (\text{NUMBUF} + \text{NUMBUFG})$$

Monitoring disk buffers

If you want to implement NUMBUFG and do more analysis later, you might start by setting NUMBUFG equal to your MAXBUF setting and leaving MAXBUF at its current setting. Then you could use the MONITOR DISKBUFF commands to analyze the buffer pool utilizations. For an example of using MONITOR DISKBUFF, see “MONITOR DISKBUFF example” on page 131.

Managing delayed disk updates

The disk update process allows delayed disk updates, which avoids duplicate database writes in certain situations.

When a user completes an update unit for a file and there are no other update units active against that file, Model 204 writes the buffer to disk with all of the file's modified pages, marks the file as physically consistent, and issues a message stating that the disk update is complete.

Specifying delayed updates—DKUPDTWT parameter

The CCAIN parameter DKUPDTWT specifies delayed disk updates. The value of DKUPDTWT determines how many seconds a disk buffer containing a file's modified pages must have aged before it can be written to disk.

When DKUPDTWT is zero, the default value, Model 204 writes all of the file's modified pages to disk at the end of the last in-flight update of the file. The user who completed the last in-flight update waits for this disk update process to complete and for the message stating that the disk update is complete.

If DKUPDTWT is not zero, Model 204 delays the start of the disk update process for at least DKUPDTWT seconds, after which it may be the checkpoint pseudo subtask (CHKPPST) that performs the disk update. The user who completed the last in-flight update does not have to wait for the disk update process to complete.

When DKUPDTWT is not zero, the CHKPPST and CHKPTIMR pseudo subtasks are started automatically at Model 204 initialization. An error message informs you if the NSUBTKS parameter is set too low to start these pseudo subtasks.

The maximum value of DKUPDTWT depends on the value of the CPTIME parameter. If CPTIME is nonzero, DKUPDTWT must be less than or equal to CPTIME*30. The absolute maximum value of DKUPDTWT is 60.

The system manager can reset DKUPDTWT to zero while the online is running. It can be reset to a nonzero value as long as the CHKPPST and CHKPTIMR pseudo subtasks were started during Model 204 initialization.

Handling delayed updates and CHKPPST

The CHKPPST pseudo subtask plays a central role in handling delayed disk updates. When DKUPDTWT is set to 0, CHKPPST does the following:

1. Sleeps for CPTIME minutes.
2. Tries to quiesce updates, for up to CPTQ, plus CPTO seconds.
3. Takes the checkpoint, if all updates are quiesced.

If DKUPDTWT is greater than 0, CHKPPST has substantially more processing to perform:

1. Sleeps for DKUPDTWT divided by four seconds.
2. Further processing depends on the value, rounded to the nearest integer of:

$$N = (\text{CPTIME} * 60) / (\text{DKUPDTWT} / 4)$$

Attempting a checkpoint

If the number of wake-up calls since the last checkpoint is N , CHKPPST takes a new checkpoint, as follows:

Attempt	Purpose
1.	Performs the disk update process on all files that are not being updated, regardless of how long since they were marked disk-update-needed.
2.	Attempts to deactivate Host Language Interface (HLI) updates for CPTQ seconds. Performs the disk update process on all files that are not being updated each time Model 204 determines that there are more HLI jobs to wait for.
3.	Attempts to deactivate all other updates for CPTO seconds. Performs the disk update process on all files that are not being updated each time Model 204 determines that there are more users to wait for.
4.	If all updates have been deactivated, performs the disk update process on all remaining files marked disk-update-needed. Otherwise, abandons the checkpoint attempt.
5.	Performs the disk update process again. Then takes the checkpoint.

Attempt	Purpose
6.	If the number of wake-up calls since the last checkpoint is not N , CHKPPST performs the disk update process on all files that have aged sufficiently—that is, marked disk-update-needed for at least DKUPDTWT seconds.

Factors affecting disk update and checkpoint processing

Several important factors affect the processing of disk updates and checkpoints:

- Some disk updates might be interrupted by another thread's request for the file's UPDATE resource. Attempts 1, 2, 3, 4, and 6 might be interrupted by an attempt to start a new update or by a user doing a disk update as part of CLOSE FILE processing. In these cases, the following message is issued, and the next file is processed:

```
M204.0440: DISK UPDATED ABORTED
```

- Attempts 2, 3, and 4 might be interrupted by the expiration of the waiting time set with Model 204 parameters CPTQ and CPTO, respectively. In these cases, the M204.0440 message is issued, all remaining files are bypassed, and the checkpoint is timed out.
- If CPTQ and CPTO are zero (no time-out intervals specified) and the DKUPDTWT parameter is nonzero, updates are deactivated for as long as required to perform the disk update process for all available files. If you do not want checkpoint attempts to deactivate updates, set DKUPDTWT to zero.
- If Attempts 2, 3, or 4 abort due to a CPTO or CPTQ time-out, the checkpoint time-out message indicates the name of the file being written at the time of the time-out:

```
M204.0843: CHECKPOINT TIMED OUT ON date/time UPDATING
FILE file
```

- Attempt 5 cannot be interrupted once it begins. Therefore, CPTO and CPTQ intervals are not honored for a CLOSE FILE command that is blocking a checkpoint. This type of event is likely to be infrequent and of short duration.
- The sleep intervals of CHKPPST are not adjusted by the amount of time required to perform Attempt 6. Therefore, checkpoints might be spaced out by more than CPTIME minutes. If this is a frequent problem, adjust CPTIME downward.

Understanding file statistics

For descriptions of the file statistics DKUPTIME, UPDPTIME, and PDNGTIME, see “Disk buffer monitor statistics and parameters” on page 435.

Handling 64-bit statistics

To support very long running Model 204 regions, Rocket Software modified the capacity of statistical counters by increasing the size of some statistics and also exploiting 64-bit processing where appropriate. For any in-house or third-party support applications that process statistical counters, you need to review the statistics generated.

As some of the statistics fields are now double-words, check *Appendix A: Using System Statistics* for the new layout of the System, Final and Partial statistics. Also, additional Disk Buffer Monitor, MP/204, and File statistics have been updated.

Look at your in-house or third-party support applications to see if you need to make changes because of the increased length of some of the statistics. Make any changes necessary to your applications, then reassemble with this new release.

If your in-house or third party support applications don't reference any of these double word statistics, then you only need to reassemble your program with the new offsets documented in this new release.

Server areas

Server areas are the internal work areas allocated to each user. Each area is divided into a fixed and variable portion. The fixed portion, which includes logical I/O buffers and user resettable parameters, is calculated by Model 204 at initialization. The variable portion can be changed dynamically with the UTABLE command (see the *Rocket Model 204 Parameter and Command Reference*) or the IFUTBL IFAM function (see the *Rocket Model 204 Host Language Interface Reference Manual*).

Sizing user server areas

The default size of all user server areas is set on User 0's parameter line. If the default is used, the allocated server area is exactly large enough to contain the tables for each user specified on each user's parameter line. If SERVSZ is also specified on a particular user's parameter line, the default is overridden for that user.

The value of SERVSZ must be as large as, or larger than, the user's aggregate table size. It is calculated by examining the user's server area requirements and monitoring the system statistics (described in "ONLINE monitoring" on page 120) that provide information about the installation work load. The following formula gives the approximate size:

$$\text{SERVSZ} = \text{fixed-table-size} + \text{variable-table-size}$$

Where:

- *fixed-table-size* represents settings, defined during initialization, which cannot be modified during the run.

- *variable-table-size* represents settings that can be varied using the UTABLE command or its IFAM equivalent, IFUTBL.

SERVSIZE and server page alignment

Servers and some server tables are always aligned on a 4K page boundary. In pre-7.4.0 releases, server and tables alignment took place only when DSPOPT had settings of bits X'01' or X'02' or the APSYPAGE parameter was indicated.

If you used server alignment previously, there is no change in your server size requirements.

If you did not use server alignment previously, then you might notice an increase in server size that in the worst case could be up to 24528 bytes per server.

When you calculate server size, take into account that FSCB, HEAP, NTBL, QTBL, STBL, and VTBL are each rounded on a 4K page boundary, so in the worst case each area could require up to 4088 bytes of server space, compared to servers with no alignment in previous releases.

The following sections explain how server area sizing parameters are processed, which parameters determine fixed and variable table sizes, and the ranges of values these parameters can take.

Initialization and error handling

During initialization, each user, except User 0, is identified in the output before the user's parameter line is read. The aggregate size of each user's tables and the size of tables fixed during initialization are printed after the user's parameters are read.

If errors are detected, they are reported and initialization continues whenever possible. If errors are detected during initialization, the run is canceled at the end of initialization. Error conditions in initializing the server cause the run to end immediately with a return code of 96.

The results of user changes to the sizes of FTBL, GTBL, ITBL, and XTBL are discussed in the *Model 204 Parameter and Command Reference*.

Calculating fixed table size

Use the following formula to calculate fixed table size, the FIXSIZE parameter value:

```
Fixed table size = 2520
                  + ((LAUDPROC + 9) * 4)dwr
                  + (LIBUFF + 4)
```

```

+ (LOBUFF + 5) dwr

+ (LOUTPB) dwr

+ ((NGROUP + 12) * (NRMTFILE + NFILES + 1))

+ (((NORQS*3) + 2) dwr + (NRMTFILE + 1)) dwr

+ (3 * (ERRMSG_L - 80))

```

Each term of this formula that is followed by *dwr* must be double word rounded to the next multiple of eight. For example, if the value of LOBUFF is 500, the term (LOBUFF + 5) = 505, which must be rounded to 512, the next multiple of 8.

If SYSOPT = 1 or 2 (indicating CCASYS or CCAGRP), add 1 to the value of NFILES used in the formula. If SYSOPT = 3 (indicating both CCASYS and CCAGRP), add 2.

If any SQL threads are specified in CCAIN (IODEVs 13, 17, or 19), add 6712 bytes for C language work areas.

Table 2-6 shows the minimum, maximum, and default values for parameters that affect fixed server table sizing. The rightmost columns show the relevant units of measure; for example, the maximum value of NORQS is 32767 entries (not bytes). The values of LIBUFF and LOBUFF may need to be increased for SQL processing. Recommended values are LIBUFF=3000 and LOBUFF=5000.

Table 2-6. Fixed server table values

Parameter	Default	Max	Bytes/entries
ERRMSG_L	80	256	Bytes
LAUDPROC	21	253	Bytes
LIBUFF	255	32767	Bytes
LOBUFF	256	32767	Bytes
LOUTPB	0	3000	Bytes
NFILES	2	16383	Entries
NGROUP	5	16383	Entries
NORQS	5	32767	Entries
NRMTFILE	0	16383	Entries

Calculating variable table size

Use the following formula to calculate variable table size:

```
Variable table size = 96
```

```

+ ((HTLEN+5) * (MAXHDR + MAXTRL)) dwr
+ (LFSCB) dwr
+ (LFTBL) dwr
+ (LGTBL) dwr
+ (LHEAP) dwr
+ (LITBL) dwr
+ (LNTBL * 12)
+ (LPDLST +32) dwr
+ (LQTBL * 16)
+ (LSTBL) dwr
+ (LTTBL * 4) dwr
+ (LVTBL * 32)
+ (LXTBL) dwr

```

Each term of this formula that is followed by dwr must be doubleword rounded to the next multiple of eight. Table 2-7 shows minimum and maximum values.

Table 2-7. Variable server table values

Parameter	Default	Max	Bytes/entries/lines
HTLEN	132	32767	Bytes
LFSCB	0	65528	Bytes
LFTBL	1000	30 million	Bytes
LGTBL	288	2 billion	Bytes
LHEAP	0	2 million	Bytes
LITBL	0	32760	Bytes
LNTBL	50	32760	12-byte entries
LPDLST	2600	32760	Bytes
LQTBL	400	262,143	16-byte entries
LSTBL	600	16M	Bytes
LTTBL	50	8190	4-byte entries
LVTBL	50	524287	32-byte entries
LXTBL	1000	32760	Bytes

Table 2-7. Variable server table values (Continued)

Parameter	Default	Max	Bytes/entries/lines
MAXHDR	5	32767	Lines
MAXTRL	5	32767	Lines

Server tables

Server tables are sections of the server area used by the User Language compiler and evaluator to store all the information necessary to run a request. Some server tables are also used by the editor and HLI functions.

Each user has a copy of the server tables in the server. Table sizes are controlled by the parameters shown in Table 2-4 on page 27. Parameter settings on the user's parameter line affect the size of the servers and the region.

Table 2-8 lists the server tables. For more information on individual tables, continue reading this chapter and see the *Rocket Model 204 User Language Manual*.

Table 2-8. Summary of server tables

Table	Contents
FSCB	Menu, screen, and image information
FTBL	File groups
GTBL	Global variables
ITBL	Dummy string and \$READ responses
NTBL	Statement labels, list names, and variables
QTBL	Statements in internal form (quadruples)
RTBL	User privileges, class, and field level security information
STBL	Character strings
TTBL	Temporary work pages
VTBL	Compiler variables
XTBL	Procedure security

Full-screen buffer table (FSCB)

The full-screen buffer table (FSCB) stores menu, screen, and image definitions, in addition to the values of screen variables and image data blocks. FSCB space is reused by each logical menu definition, logical screen definition, or block definition.

The FSCB must be large enough to hold the largest screen, image, or menu definition. The following space is required:

- 144 bytes of fixed overhead for every menu, including the menu title
- 144 bytes for each menu prompt
- 432 bytes of fixed overhead for the first panel of every screen:
 - 144 bytes for each subsequent panel, including the screen title
 - 32 bytes for each screen prompt and input item
 - 32 bytes for every screen line containing at least one input item
 - 80 bytes for each defined screen line, including skipped lines
- Additional space for automatic validation, including:
 - 2 or 4 bytes for each automatic validation option
 - 256 bytes for the VERIFY command when a particular character set is used in a compiled screen for the first time

Additional occurrences of the same character set do not add extra space. ONEOF and character RANGE store each character string plus one byte for each string's length.
 - 8 bytes for each number in a NUMERIC RANGE statement (16 bytes for each range pair)
- Space for every block used in image definition

The amount of required space is computed as the sum of the specific User Language statements, clauses, and items. For a complete list of these values, refer to the *Model 204 User Language Manual*.

File group table (FTBL)

Data structures particular to file groups are stored in FTBL. FTBL entries are:

- Sixty-two-byte fixed size entry plus six bytes for each file in the group definition

This entry is allocated each time a group is opened (explicitly by the OPEN command or implicitly for an ad hoc group) and is released when the group is closed.
- Variable entry consisting of nine fixed-bytes plus a number of bytes equal to the length of the field name plus 11 bytes per file in the group

This entry is created for collecting field-name codes and properties during a User Language request. An entry is allocated each time a new field name is encountered in the request. The field entries are deleted at every END statement (including END MORE).

When the Inverted File Access Method (IFAM) is used:

- Host Language threads use FTBL under the same circumstances as User Language.
- Field entries are not deleted until the group is closed or until IFFNSH is called.
- Increase the total FTBL requirement by NGROUP times four bytes.

For Parallel Query Option/204 server nodes, the required size of FTBL increases by eleven bytes times the total number of group members for temporary scattered groups opened at the client and containing a member on the server node.

For information on using FTBL in 64-bit storage, see “FTBL and above the bar storage” on page 172.

Understanding the global variable table (GTBL)

GTBL contains information about:

- Global variables
- Global images, screens, and menus

GTBL entries are created by the \$INCRG, \$GETG, and \$SETG functions. When a global variable is redefined, its old entry is deleted from GTBL and a new entry is added.

In addition, a 32-byte trailer stores information about offsets.

Clearing GTBL entries

The CLEARGO command deletes all images, screens, and menus. You can also use the CLEAR GLOBALS statement to delete selected types of GTBL entries. For details, see the *Rocket Model 204 User Language Manual*.

Space allocation

The space allocation for a global variable includes:

- 4 bytes indicating the length of GTBL
- 1 byte for the length of the variable name
- Variable name
- 1 byte for the length of the current name
- Current value

Global images, screens, and menus require space for a 20-byte header in addition to the size of the object.

When allocating GTBL space, always remember to add 32 bytes for the trailer.

The minimum length of GTBL is 40 bytes (X'28').

Improving global variable processing

You can improve global variable processing by setting the FASTGLOB, GLBLPCT, and GTBLHASH parameters. See the *Rocket Model 204 Parameter and Command Reference* for a description of the possible settings and the *Model 204 User Language Manual* in the chapter on global features for a discussion of how these parameters affect performance.

Dummy string and \$READ table (ITBL)

ITBL holds dummy string and \$READ responses that are entered as arguments to an INCLUDE statement or command.

The space allocation for an ITBL entry includes:

- Argument strings, including delimiters, which are saved as they are entered
- 4 bytes of overhead for each saved string

Space taken by a string is released when the included procedure is executed.

Labels, names, and variables table (NTBL)

NTBL holds labels, names, and variables. Each entry is allocated 12 bytes. NTBL has two entries for each first occurrence of a COMMON declaration.

NTBL has one entry for each of the following elements:

- Statement label
- List name
- %variable
- Image, menu, and screen variable
- Partner process opened by a request
- Additional COMMON declaration
- Unlabeled FIND
- FOR EACH VALUE statement
- FOR statement with the IN ORDER clause
- Sequential or VSAM file opened simultaneously

Most NTBL entries are preserved by the MORE command except for the unlabeled FIND and secondary FOR entries, which are deleted.

A host language thread requires NTBL entries for list names, compilation names, and variables.

FLOD uses NTBL entries for tags and index registers. The size of NTBL determines the highest tag or index that can be specified. In this case, NTBL must be at least 12 bytes multiplied by the highest tag or index size desired.

You need not allow extra space for runtime NTBL storage used during request evaluation of an OPEN DATASET, OPEN EXTERNAL, or OPEN TERMINAL statement. Use the compiler high-water mark to set LNTBL.

Internal statement table (QTBL)

QTBL holds internal Model 204 instructions that result from compilation of each internal statement. After compilation, the entries in QTBL drive the evaluator. QTBL is emptied by END and END MORE statements.

The Editor formats all of QTBL into 16-byte entries, which provide a map of text being edited. The number of entries used depends on the number and position of insertions and deletions, not on the amount of text.

Quadruple (QTBL) entries generated by User Language statements vary in number and size. Table 2-9 on page 55 shows typical values for each entry. For more information, refer to the *Model 204 User Language Manual*.

You can reduce server I/O by allowing users executing shared precompiled procedures to use a shared copy of QTBL.

QTBL and IFAM processing

When using the Inverted File Access Method (IFAM):

- IFFIND, IFCOUNT, and list manipulations build quadruples so that the User Language evaluator routines can be used.
- IFGET, IFMORE, and IFPUT build field name lists in QTBL.
- Other calls are evaluated directly.

IFAM's use of QTBL and the effect of the compiled IFAM feature are discussed in detail in the *Model 204 Host Language Interface Reference Manual*.

QTBL and FLOD processing

FLOD builds its own quadruples (flodruples) in QTBL. Flodruple sizes vary, but most are 20 bytes or less.

Major exceptions are:

- Read-and-load-field flodruple, which expands four bytes for each entry in a translation table

- CASE statement, which requires eight bytes for each comparison string

Sample QTBL entries

Table 2-9 shows typical QTBL entry sizes for various User Language statements and program structures.

Table 2-9. Sample QTBL entries

User Language statement	QTBL entry size
\$functions	16 + 3 per argument
%variable, subscripted (reference to)	16 + expression evaluation
ADD	20
AFTER	20
AND (except where AND is part of BETWEEN)	16
Conversion between a string and a number	16
CALL	16
CHANGE	44
CLEAR LIST	20
CLEAR ON	16
CLEAR TAG	16
CLOSE	16
CLOSE PROCESS	16
COMMIT	4
COMMIT RELEASE	20
COUNT RECORDS (with a group)	52 + 20
DELETE	24
DELETE RECORD	16
ELSE	16 + body of the clause
ELSEIF	16 + body of the clause END
END	4
Function call	16 + argument evaluation
FIND (with at least one direct condition)	64 + 16
FIND (with inverted condition)	64 + 36

Table 2-9. Sample QTBL entries (Continued)

User Language statement	QTBL entry size
FIND ALL RECORDS (no qualification)	64
FIND ALL RECORDS (with record security and group)	64 + 52 + 20
FIND ALL RECORDS (with record security)	64 + 52
FIND ALL VALUES (FRV field)	74
FIND ALL VALUES (ORDERED field)	32
FOR EACH RECORD	24 + body of the loop
FOR EACH VALUE OF (with IN ORDER, a group)	88 + body of the loop + 68 + 24
GREATER THAN	20
IDENTIFY	16
IF	32
IF (with operator)	32 + 16
Index loop	40 + expression evaluation + body of the loop
IN RANGE FROM and TO, BETWEEN	28
INSERT	24
MODIFY	16
NOTE	20
ON	16 + included statements
OPEN	20
OPEN PROCESS	20
OR	16
PAUSE	16
POSITION	20
PREPARE IMAGE	8
PREPARE MENU	8
PREPARE SCREEN	8
PRINT	16 for each term
PRINT (a field)	16 + 20
PRINT MENU	16

Table 2-9. Sample QTBL entries (Continued)

User Language statement	QTBL entry size
PRINT SCREEN	16
READ IMAGE	16
READ MENU	20
READ SCREEN	20
RECEIVE	16
RELEASE RECORD	20
RELEASE POSITION	8
REPEAT	16 + evaluation of WHILE clause
REREAD SCREEN	20
RETRY	16
RETURN	16
SEND	16
SIGNAL PROCESS	12
STOP (automatically generated)	16
STORE RECORD (with each field)	16 + 16
TAB	4
TAG	16
THEN	16 + body of the clause
TRANSFER	16
WITH	0
WRITE IMAGE	12

User, field, group security table (RTBL)

RTBL contains a user's privileges, class, field-level security (FLS) levels for each open file, and classes for open, permanent groups.

The size of RTBL is calculated from the formula:

$$((\text{NGROUPS} + 11) * (\text{NFILES} + 1)) + \text{NRMTFILE} + 1$$

For Parallel Query Option/204 client nodes, the required size of RTBL increases by (NRMTFILE=1) bytes.

Character string table (STBL)

STBL stores all character strings in counted form, with a 1-byte length preceding the string itself. The following considerations apply to space usage:

- Space used to store intermediate results during an arithmetic expression evaluation is freed when the evaluation is completed.
- Space used by FOR EACH OCCURRENCE, FOR EACH RECORD, and FOR EACH VALUE loops is reused until the end of the loop.
- The last value of a NOTE statement remains in STBL.
- FIXED or FLOAT %variable array uses eight bytes for each element.
When the %variable is reassigned, the STBL space is reused.
- The FIELD SAVE option requires 10 bytes plus the maximum length of the string plus one byte for each element.
NO FIELD SAVE does not reserve the extra 10 bytes and results in significant saving when using a multidimensional array.
- MORE releases all but the space required by %variables and arrays.
- If a user is using the pattern matcher in an IF statement with an IMAGE or SCREEN ITEM as the pattern and an IMAGE or SCREEN ITEM as the comparison string, then the value of the pattern is stored in STBL. The space is freed after each comparison, so the maximum increase is equal to the size of the largest pattern in a request that meets the above criteria.

When the Inverted File Access Method (IFAM) is used

- IFDVAL and IFFILE store the value string from the input parameter in STBL.
- Strings enclosed in quotation marks or values specified for IFFIND that are stored in STBL are the same types as those stored in STBL by User Language.
- EDIT form of IFPUT uses STBL for each value. Space from previous values is reused.
- FLOD stores translation table values and CASE statement comparison values in STBL.

Parallel Query Option/204 STBL requirements

STBL requires an increase for Parallel Query Option/204 \$functions used in remote file or scattered context and for User Language pattern matching processes

- Using one of the following \$functions in remote context requires 12 additional bytes in STBL:

\$CURFILE

\$FLCFILE

\$UPDATE

\$UPDFILE

- If you are using the pattern matcher in any User Language statement, the full size of the pattern is used and stored in STBL. The space is freed after each statement block using the pattern matcher, for example, FIND or FOR.

Temporary work page list table (TTBL)

TTBL entries keep track of scratch file (CCATEMP) pages. The entries are four bytes each and used by the Editor and FIND (or IFFIND) evaluator routines.

- The Editor uses scratch pages to make a private copy of the procedure being edited.
- FIND uses scratch pages as work space for evaluating Boolean expressions.

The number of TTBL entries required by FIND depends on the complexity of the Boolean expression. Entries are released at the end of the FIND statement evaluation.

Compiler variable table (VTBL)

VTBL stores values of simple variable types and refers to the String Table (STBL) in cases of more complex variables. The variables are local (%variables), internal, screen, and image variables.

Many User Language statements and some constructs cause one or more compiler variables to be allocated in VTBL.

Entries in VTBL vary in size. Table 2-10 shows typical values. For more information, refer to the *Model 204 User Language Manual*.

Table 2-10. Sample VTBL entries

User Language statement	VTBL entry size (bytes)
% variable:	
FIXED	16
FLOAT	12
STRING	24
array	20
array element reference	12

Table 2-10. Sample VTBL entries (Continued)

User Language statement	VTBL entry size (bytes)
CALL	4 + (4 * arguments)
CALL statement evaluation	28
COUNT	8
FIND (basic entry, single file)	8
workspace	20 + 20 + 28
fieldname = value pair	20 or more
retrieval	28
Ordered Index retrieval	4 + 4 per segment
FIND (basic entry, group)	8 + (8 * files)
FOR EACH RECORD (no IN ORDER clause)	16
FOR EACH VALUE (with FROM, TO, LIKE, ORDERED)	48
FOR EACH RECORD IN ORDER BY (with ORDERED field)	48
FOR EACH VALUE (with ORDERED field)	40
Function (User Language)	4 + (4 * arguments)
Lists:	
single files	8
groups	8 + (8 * files)
Menu definition	48
Numeric expression	16
ON	28
Related image set	12 + 4 for every 256 items
Screen definition	68 + 4 for each panel
Screen entry (one panel)	72
SORT	
each referenced field	20
each sort key	32
String expression	8
Subroutine declaration	16 + parameters

Effect of the MORE parameter on VTBL

The MORE parameter deletes all VTBL entries except %variables, list entries, images, menus, screens, labeled SORT and sort key entries, labeled FIND entries, and labeled COUNT entries.

IFAM and FLOD considerations

The following considerations apply to VTBL:

IFAM requires VTBL space for IFFIND, IFCOUNT, and lists (matching User Language requirements), and a few extra bytes for temporary work space.

FLOD needs a minimal amount of VTBL space if the L statement (locate loop) is used.

VTBL requirement for Application Subsystem users

The size of VTBL for Application Subsystem users loading saved compilations does not have to be as large as the compiling user's VTBL. As long as the loading user's VTBL size accommodates the request's VTBL requirement, the loading user can have a smaller VTBL than the compiling user.

Procedure security table (XTBL)

XTBL contains procedure security information for each file and permanent group member that a user has open. The size of each file entry depends on the number of user-class/procedure-class mappings defined in the file for the user's class.

3

Defining the User Environment (CCAIN)

In this chapter

- Overview
- Setting user parameters
- Access methods and device types
- BSAM (IODEV=3)
- SNA Communications Server terminal support (IODEV=7, 37)
- CRAM (IODEV=11, 23, 29)
- CRAM IFSTRT protocol IFAM2 (IODEV=23) requirements
- SQL server threads (IODEV=19)
- Horizon (IODEV=27)
- IUCV (IODEV=39, 41, 43)
- RCL thread (IODEV=49)
- M204 command (CMS)
- CMS service machine console (ALTIODEV=45, 47)
- User environment control parameters

Overview

This chapter summarizes the access methods and device types compatible with Model 204 and explains special considerations relating to operating systems and Model 204 configurations. For detailed information about device types, refer to the *Model 204 Terminal User's Guide*.

The most common parameters used to control a user's environment are summarized at the end of this chapter. See the *Rocket Model 204 Parameter and Command Reference* for details about the full range of individual user parameters.

Setting user parameters

Specifications controlling the environment of each Online user are entered in the CCAIN input stream on user parameter lines following User 0's runtime specifications. Each user line defines the input/output access method and device (IODEV parameter), followed by appropriate control parameters.

Basic rules

The following rules apply:

- Define each user to Model 204 on a separate parameter line following User 0's parameter line.
- Number of user parameter lines must be one less than the value specified for the number of users (NUSERS - 1).
- You must define terminals of the same type together.
- Any user parameters set on User 0's parameter line take effect for current users unless they are specifically reset.
- Initial setting of a parameter on one line remains in effect until a second parameter is read that explicitly resets the first. If no parameter needs to be set (because all previous parameter settings are applicable), you can use a statement with a single asterisk.
- Terminal interface is initialized if one or more IODEV statements of the same type are encountered in CCAIN.

Access methods and device types

Each Online user is defined to Model 204 on a separate parameter line that first defines the user's device type and access method (IODEV parameter) and then specifies other parameters as required. The IODEV definition provides a value that is used in the context of a table lookup to obtain current routines that handle that particular device or access method.

Access methods

Model 204 supports the following access methods:

Access method	Acronym
Basic Sequential Access Method	BSAM
Cross-Region Access Method	CRAM
Inter-User Communication Vehicle	IUCV
Remote Command Line	RCL
Virtual Telecommunications Access Method	SNA Communications Server (formerly VTAM)

Table 3-1 and Table 3-2 list each access method with its associated IODEV setting and required system (User 0) and user parameters. Details on each IODEV setting follow.

Device types (IODEV settings)

Table 3-1 lists z/OS and z/VSE input/output device types with required parameters.

Table 3-1. z/OS and z/VSE device type codes

IODEV value	Access method/device type	Essential parameters
1	CCAIN/CCAPRINT (User 0 only)	(none)
3	BSAM	INPUT OUTPUT
7	SNA Communications Server 3270 (full screen)	LOUTPB MODEL NOTERM NOUTBUF NSUBTKS TERMBUF TERMOPT VTAMNAME
11	CRAM thread (full screen), supports: <ul style="list-style-type: none"> • TSO • CICS • Program Communication facilities 	CRFSCHNL LOUTPB MODEL NOTERM POLLNO

Table 3-1. z/OS and z/VSE device type codes (Continued)

IODEV value	Access method/device type	Essential parameters
19	Horizon SQL thread	INMRL NOTHREAD NSUBTKS SQLBUFSZ
23	CRAM, Host Language IFAM2 thread	IFAMCHNL
27	Horizon	NOTERM
29	CRAM, User Language thread (line-at-a-time) z/OS and z/VSE, supports: <ul style="list-style-type: none"> • CICS • TSO 	CRIOCHNL NCCC INMRL NOTERM OUTCCC OUTMRL POLLNO
37	SNA Communications Server 3767 and NTO (line-at-a-time)	INCCC INMRL NOTERM NSUBTKS OUTCCC OUTLPP OUTMRL TERMID TERMOPT VTAMNTO
49	Remote Command Line (RCL)	NOTHREAD POLLNO

Table 3-2 lists z/VM/CMS settings for the IODEV and ALTIODEV parameters.

Table 3-2. z/VM input/output device type codes

IODEV value	Access method/device type	Essential parameters
3	BSAM	INPUT OUTPUT

Table 3-2. z/VM input/output device type codes (Continued)

IODEV value	Access method/device type	Essential parameters
7	SNA Communications Server 3270 (full screen)	LOUTPB MODEL NOTERM NOUTBUF NSUBTKS TERMBUF TERMID TERMOPT VTAMNAME
39	IUCV User Language thread (line-at-a-time)	INCCC INMRL NOTERM OUTCCC OUTMRL POLLNO TERMID VMIOCHNL
41	IUCV 3270 thread (full screen), supports Program Communication facilities	LOUTPB NOTERM POLLNO, TERMID VMFSCHNL
43	IUCV IFAM2 thread	NOTERM POLLNO TERMID VMIFCHNL
ALTIODEV=45	Single-user line mode thread (Service machine console)	
ALTIODEV=47	Single-user full-screen mode thread (Service machine console)	

The following sections provide details on each IODEV setting.

Note: z/VM/CMS system managers: take note of the IUCV section, which explains IODEV settings 41 and 43 and also discusses Model 204 command options.

BSAM (IODEV=3)

IODEV=3 manages sequential input and output from BSAM. You can configure unattended terminal simulations with IODEV=3, because BSAM assigns buffer

allocation, blocking and unblocking, and scheduling I/O services to the user. IODEV=3 uses the same internal routines that handle User 0 statements.

Effective use of IODEV=3 requires consideration of the beginning and end of processing. When using IODEV=3, Model 204 commands and requests are read from the sequential data set. Command processing begins as soon as IODEV=3 initialization is complete and continues until end-of-file is reached. At this point, the thread cannot be reactivated and processing ends.

Commands and User Language requests are read line-at-a-time from the input. Results of commands and print statements are issued to the output data set.

Uses of IODEV=3

IODEV=3 performs the following functions:

Function	Explanation
Stress test	Emulates full-screen or line-by-line users as if the input and output were coming from terminals or TP access methods.
Job scheduler	Schedules jobs to be done after a specified wait, at a specific time, or repeated at specified time intervals.
Triggers	Monitors conditions in the DBMS and invokes appropriate action when certain conditions are detected.

The following User Language program illustrates how to use IODEV=3 as a trigger. After the example, details on data set definitions and buffer allocation are presented.

Trigger example

The following example uses IODEV=3 as a trigger to monitor the DBMS.

```
//I0301I DD *
LOGON system-manager-id
password
*SLEEP 60          required to complete nucleus initializa-
tion
                    before the start of thread execution
DEFINE DATASET TXFILE WITH SCOPE=SYSTEM SEQUENTIAL SAM    -
                    RECFM=VB LRECL=137 BLKSIZE=1600
OPEN DAILY
BEGIN
IMAGE TXT
TEXT.LINE IS STRING LEN UNKNOWN
END IMAGE
FD:   FD RECTYPE=CONTROL
      END FIND
      FR FD
      IF GO='STOP' THEN STOP
      ELSEIF GO='PROCESS' THEN %GO='GO'
```

```

        ELSE %GO='PAUSE'
    END IF
    END FOR
    RELEASE RECORDS IN FD
    IF %GO NE 'GO' THEN
        PAUSE 60
        JUMP TO FD
    END IF
O2:   OPEN DATASET TXFILE FOR INPUT
R1:   PREPARE IMAGE TXT
      READ IMAGE TXT FROM TXFILE
      IF $STATUS=1 THEN
          JUMP TO FD2
      ELSEIF $STATUS=GT1 THEN
          PRINT $ERRMSG
          STOP
      END IF
      IDENTIFY %TXT:TEXT.LINE LEN %TXT@READLEN
      PRINT %TXT@TEXT.LINE
      JUMP TO R1
FD2:  CLOSE DATASET TXFILE
FD1:  FDR RECTYPE=CONTROL
      END FIND
      FR FD1
      CHANGE GO TO 'PAUSE'
    END FOR
    CMMTRL
    JUMP TO FD
END
LOGOUT
/*
//I03010 DD SYSOUT=A
. .
//

```

Input or output data set definitions

Each user defined as an IODEV=3 thread must have separate input and output data sets defined. Define input and output data sets using the INPUT=ddname and OUTPUT=ddname parameters on each IODEV=3 statement.

The IODEV statement must define INPUT and OUTPUT (summarized in Table 3-11 on page 105). The definitions must coincide with the input and output *filenames* (the DTF names) in the JCL.

Examples of data set definitions follow for z/VSE, z/OS, and z/VM environments.

z/VSE JCL example

The following JCL defines input and output data sets in a z/VSE environment:

```
// DLBL I0301I,'dataset name'
// EXTENT SYSnnn,balance of extent data
// ASSGN SYSnnn,cuu
// DLBL I0301O,'dataset name'
// EXTENT SYSnnn,balance of extent data
// ASSGN SYSnnn,cuu
** The CCAIN definitions remain the same **
IODEV=3,INPUT=I0301I,OUTPUT=I0301O
```

z/OS JCL examples

The following examples show how to define input and output data sets for IODEV=3 in a z/OS environment. Input and output DD statement names are first defined as files in the JCL and then referenced on the IODEV=3 INPUT and OUTPUT parameters.

The first example uses user-defined names that are coded on the IODEV=3 statement:

```
//I0301I DD DSN=dataset name,DISP=SHR
//I0301O DD SYSOUT=A
//I0302I DD *
LOGON system-manager-or-system-administrator-id
password

*SLEEP 60 ---- used to control the start of thread execu-
tion
INCLUDE procname
//I0302O DD DSN=dataset name,DISP=SHR
.
.
.
//CCAIN DD *
.
.
.
IODEV=3,INPUT=I0301I,OUTPUT=I0301O
IODEV=3,INPUT=I0302I,OUTPUT=I0302O
```

z/VM FILEDEF example

The following FILEDEFs define input and output data sets in a z/VM/CMS environment:

```
FILEDEF I0301I DISK TEST IN A
FILEDEF I0301O DISK TEST OUT A
```

If input is read from the virtual card reader, use:

```
FILEDEF I0301I READER
```

If output is sent to the virtual reader of another machine through the virtual punch, use:

```
CP SPOOL PUN some-user
```



```
FILEDEF I03010 PUNCH
```

Buffer allocation

Each IODEV=3 requires allocation of a set of buffers. Buffer allocation is based upon the INMRL value for input and OUTMRL for output, unless DCB information is specified in the JCL. By default, INMRL is 80 bytes and OUTMRL is 132 bytes. (See Table 3-11 on page 105 for parameter summaries.)

If DCB information is specified, the buffer allocated is based upon the stated BLKSIZE parameter. DCB information overrides INMRL or OUTMRL values.

If INMRL or OUTMRL is set on an IODEV line above the first IODEV=3, or if they are set in User 0 definitions, the INMRL and OUTMRL values define the values for IODEV=3.

The following example sets buffers of 256 bytes for both input and output:

```
IODEV=29 , POLLNO=1 , INMRL=256 , OUTMRL=256
IODEV=3 , INPUT=I0301I , OUTPUT=I0301O
```

Dynamic allocation and DFSMS/HSM

Model 204 verifies that DFSMS/HSM is active before attempting to recall a migrated data set. If DFSMS/HSM is not active and a data set is migrated, the ALLOCATE (or DEFINE) command fails.

Model 204 also detects archived data sets (volser=ARCHIV is used by non-IBM data management products). Archived data sets must be recalled before the Model 204 Online step begins. An attempt to dynamically allocate an archived data set fails.

If a dynamic allocation attempt fails in either of these circumstances, the following messages appear:

```
M204.2501: CHECK FOR DFHSM ACTIVE, RETURN CODE = %C, REASON CODE = %C
M204.2502: DFHSM RECALL ERROR, DSNAME = %C, RETURN CODE = %C, REASON CODE = %C
```

The restrictions on archived data sets do not apply to BATCH204 jobs.

To do these checks, Model 204 uses a special data set name that must not be defined in the z/OS catalog. The special data set is:

```
DSNAME='MODEL204.DUMMY.DATASET.THAT.DOES.NOT.EXIST'
```

SNA Communications Server terminal support (IODEV=7, 37)

The SNA Communications Server terminals that can log on directly to Model 204 are:

IODEV	SNA Communications Server terminals
IODEV=7	IBM 3270s and other 3270-type terminals
IODEV=37	IBM 3767s and line-at-a-time terminals supported by means of the IBM Network Terminal Option (2741s, teletypes, and teletype-compatibles).

SNA Communications Server is a fully functional component of Model 204. See the installation guide for your operating system. Verify that other terminals compatible with SNA Communications Server operation are fully compatible with Model 204 before using them.

Model 204 supports full-screen SNA Communications Server terminals that do not use the standard 3270 data stream by supplying a mechanism for writing exit routines to perform data conversion. The rules for such routines are provided in “Rules governing data conversion exit routines” on page 603.

SNA Communications Server terminals are supported in z/OS, z/VSE, and CMS environments. For direct SNA Communications Server terminal support in a CMS environment, the optional CMS/SNA Communications Server Interface feature must be installed. Only full-screen terminals (IODEV=7) are supported by the CMS/SNA Communications Server Interface, which is described on page 3-74.

SNA Communications Server network definition requirements

An APPL statement in VTAMLST for each IODEV type (IODEV=7, IODEV=37) is required for direct SNA Communications Server terminal support by Model 204. The 1- to 8- character APPL names are used as the values for the VTAMNAME (full-screen) and VTAMNTO (line-at-a-time) parameters within the User 0 CCAIN lines.

Model 204 has no further requirements regarding the APPL statements or other network definition statements. For example, network concerns alone determine the setting of logmode table entries.

Note: Model 204 supports both definite response and exception response protocols on messages outbound to the terminal. Requests for SNA definite responses to messages coming inbound from the terminal are not supported. For an exception response protocol on outbound messages, set the '02' value of the TERMOPT parameter setting on each IODEV=7 statement that refers to the terminal.

IP address using TN3270 connection to VTAM session: IPADDR

TN3270 connections via Telnet can be made to a Model 204 job with IODEV=7 connections defined in the job. For such a connection it may be desirable to know the IP address of that user's VTAM session. This information is now available via the IPADDR parameter. You can retrieve the IP address using the:

- VIEW command

```
VIEW IPADDR
```

- User Language \$VIEW function

```
%IPADDR=$VIEW('IPADDR')
```

CCAIN requirements

The following CCAIN parameters are relevant to IODEV=7 terminals:

Systemwide parameters	Per-user parameters
LOUTPB	IODEV
NOUTBUF	MODEL
TERMBUF	TERMOPT
VTAMNAME	TERMID
NSUBTSKS	IPADDR
NOTERM	

The following CCAIN parameters are relevant to IODEV=37 terminals:

Systemwide parameters	Per-user parameters
VTAMNTO	IODEV
NSUBTSKS	INMRL
NOTERM	INCC
	OUTMRL
	OUTCC
	OUTLPP
	TERMOPT
	TERMID

IODEV=7 terminals under CMS require an additional systemwide parameter VTGCSSRV (see the next section for details).

All CCAIN parameters relevant to each IODEV type are listed in Table 3-1 on page 65 and Table 3-2 on page 66. The significance of each parameter is explained in full detail in the *Model 204 Parameter and Command Reference*.

CMS/SNA Communications Server Interface for full-screen terminals

Model 204 under CMS implements its own SNA Communications Server communications requests using the CMS/SNA Communications Server Interface:

- All SNA Communications Server terminals logged on to Model 204 are driven directly by the ONLINE, as in a z/OS or z/VSE environment. The ONLINE then becomes a SNA Communications Server APPL and an APPL statement is required for Model 204 in the network's VTAMLST files.
- An additional systemwide CCAIN parameter, VTGCSSRV, is required. The CMS/SNA Communications Server Interface, although using the same SNA Communications Server terminal handler software module as in a z/OS environment (VT75), employs further software, which runs under z/VM's Group Control System (GCS). See the *Rocket Model 204 z/VM Installation Guide* for full details about setting up the CMS/SNA Communications Server Interface.
- Using the CMS/SNA Communications Server Interface for terminal support means that users are logged on to the ONLINE machine through SNA Communications Server services. A virtual machine is not required for each user.

The CMS/SNA Communications Server Interface is an optional feature. To use the Model 204 distributed application facility, Horizon, under CMS, however, the CMS/SNA Communications Server Interface must be installed. For further information about Horizon, see the section "Managing IODEV=27 threads" on page 97.

- IODEV=37 terminals are not supported by the CMS/SNA Communications Server Interface.
- IODEV=41 support for full-screen terminals under CMS remains available whether or not the CMS/SNA Communications Server Interface is installed.

CRAM (IODEV=11, 23, 29)

The Cross-Region Access Method (CRAM) is an interregional facility that lets applications access Model 204. The access is defined by device type codes and channel names in User 0's CCAIN stream. (See Table 3-5 on page 83.)

- For IFAM applications, communication handled by CRAM is a two-way conversation.
- For non-IFAM applications, communication handled by CRAM involves one program as the master and the other as a user. The master is always the Model 204 service program, which can communicate with multiple users

over the same channel. After a connection is established, the user thread can issue requests, as required, for the application.

Remote User Language (IODEV=11, 29) and IFSTRT protocol IFAM2 (IODEV=23) threads require the use of the CRAM. Terminals are considered remote User Language threads when either the CICS or TSO is used. (See the *Rocket Model 204 Terminal User's Guide* for information about these interfaces.)

Facilities requiring CRAM

You must install CRAM if you use the following products and interfaces. In the following table, the products are grouped according to IODEV value:

Facilities	IODEV value
Full-screen support for: <ul style="list-style-type: none"> • CICS • TSO 	11
SQL thread	13
Host Language IFAM2/IFSTART	23
Line-at-a-time User Language thread for: <ul style="list-style-type: none"> • BATCH2 • CICS • IFAM2/IFDIAL • INTERCOMM • TSO 	29

CRAM options for z/OS

On z/OS sites, Model 204 provides two CRAM options. You can choose the CRAM option by setting the XMEMOPT in User 0's CCAIN stream. z/VM and z/VSE sites have only one CRAM option.

You can run the following CRAM options on the same machine. A Model 204 Online, however, can operate only one CRAM option at a time.

CRAM	The Cross-Region Access Method works as it has always worked for both z/OS and z/VSE.
Cross-Memory Data Mover (CRAM-XDM)	<p>CRAM-XDM initializes only one cross-memory environment that is shared among all Onlines. Submit the M204XDM (XDM master) job first. For Onlines that use CRAM-XDM, the XDM master must be running.</p> <p>XDM address space is nonswappable and uses a system Linkage Index (LX). The address space is noncancelable and always recovers from an abend.</p>

Because only one Subsystem Access Control Block (SSACB) is established per z/OS image, XDM always reuses the same system LX, even if forced down. A new address space is always used, because a system LX always makes the address space nonreusable. (However, XDM must be started and left active for the life of the IPL.)

XDM terminates, on request, only if all cross-memory connections have been disconnected. An outstanding operator reply is used to communicate to XDM. In addition to terminate, a MONITOR,ONLINES command lists on the operator console all address spaces that have connections to XDM. All data moves are done using access registers.

Choosing the CRAM option to use

After Model 204 and CRAM are installed, the Model 204 installer can choose which CRAM option to associate with a Model 204 Online.

Model 204 now supports the use of cross-memory services within CRAM under z/OS/ESA and higher. You can expect the following performance enhancements when you choose to use CRAM-XDM:

Performance enhancement	Result
CRAM buffers are no longer obtained from CSA.	Substantial decrease in CRAM CSA storage requirements.
All data moves are from the user buffer directly into the Model 204 user area.	Significant decrease in the number of data moves.
CRAM waits under cross-memory services are invisible.	No ECBS are added to the scheduler chain, which reduces the length of the Model 204 wait chain, thus reducing the evident load on the Online scheduling process.

XDM improved ALET (Access List Entry Token) capacity

The XDM facility allows the full use of ALETs up to the IBM limit of 512.

This allows you to support more concurrent connections between multiple ONLINES and multiple XDM clients connecting to those ONLINES (CICS tasks and batch IFAM tasks).

For example: In previous releases, a single CICS region that communicated to 3 ONLINE regions would need 3 ALETs, one for each ONLINE. In this release, each CICS region may talk to multiple ONLINES using one ALET. So for 3 CICS regions to communicate to 3 ONLINE regions, a total of 6 ALETs is needed – one ALET for each CICS region and one for each Model 204 ONLINE.

You might choose to use CRAM at your site because:

- Your site does not allow nonswappable, APF authorized tasks in your environment.

- CRAM performance is a non-issue.
- CSA storage limitations are a non-issue.

Invoking a CRAM option

Onlines

The CRAM option invoked is determined by setting the XMEMOPT parameter in the CCAIN input.

This XMEMOPT setting...	Initializes...
X'80'	CRAM-XDM environment
Any other setting	CRAM environment, if CRAM threads are defined.

CRAM-XDM Onlines are compatible with the CRAM option; both can be used on the same z/OS subsystem. However, a given address space can use only one method. All channels within the Online address space can use either the CRAM-XDM option or the CRAM option, but not both.

M204XDM

A system LX is intended for a long-running task and must be used carefully because it is a limited system resource. The address space that uses a system LX is, by definition, not reusable.

Warning: If this parameter is not used carefully, a shortage of address spaces can occur, resulting in a system IPL. The use of system LXs is discussed in z/OS licensed manuals.

For Online and applications

Nonswappable address space is not required. If an address space is swappable, performance degrades slightly due to the necessity of resuming an SRB and the z/OS scheduling delays that entails. Nonswappable address spaces require no SRBs and thus no z/OS scheduling delays. Therefore, the fastest environment involves:

- Nonswappable application
- Nonswappable Online
- CRAM-XDM

Implementing CRAM XDM usage for z/OS operating systems

To implement CRAM-XDM, take the following steps:

1. Install Model 204 following the instructions in the *Model 204 z/OS Installation Guide*.
2. You must include a PPT entry for the M204XDM module. Specifying non-swappable is mandatory and non cancelable is recommended.
3. In the M204XDM job, set the PARM= parameter on the EXEC statement to the subsystem name used in the CRAMINS installation job, which assembles IGCLM244.
4. You must submit the M204XDM job prior to all Onlines. If M204XDM is unavailable and the Online specifies XMEMOPT=X'80', the Online cannot initialize.
5. Set the appropriate XMEMOPT options in your Model 204 Online CCAIN stream. To use CRAM XDM, include X'80' in the XMEMOPT setting and set XMEMSVC to the SVC number used in the XSVCINS installation job. If you linked M204XSVC into your Online instead of installing it as an SVC, do not set XMEMSVC.

Warning: If you make CRAM XDM cancelable and the operator cancels Model 204 XDM, CSA storage is orphaned. **To reclaim the orphaned storage, you must IPL z/OS.**

SNAPCRAM utility

The SNAPCRAM utility supports both CRAM options, so that Technical Support can diagnose problems for either option.

Cross-memory debugging facility

A cross-memory debugging facility is available for CRAM-XDM. The TRACEX command saves the status of each XDM call in a user wrap-around trace table. Use the trace facility only when Model 204 Customer Support requests it.

TRACEX is valid only for XDM from the address space supporting the given channel. If XDM is not active, the message "cross memory inactive" is displayed. Use the following command to invoke a trace:

Syntax `TRACEX channel action (parameter)`

- Where**
- *channel* is the name of the channel to be traced
 - *action* is one of the following *parameters*:

Parameter	Action
INIT (<i>size</i>)	Creates a wrap-around trace table of (<i>size</i>) size
START (<i>user</i>)	Traces a user (default = all)
STOP (<i>user</i>)	Stops tracing a user (default = all)

Parameter	Action
CLEAR	Frees the trace table
VIEW	Shows trace table address and trace status

Monitoring XDM

After CRAM is installed and the M204XDM job is up and running, you can monitor CRAM-XDM, either from the console or from a batch job.

Using console commands

To monitor XDM, you can issue a MONITOR command. (Authorized users can also issue console commands from SDSF using the */nn* syntax). When XDM master address space is active, it displays the following reply message on the operator console:

Syntax

**nn M204XDM.100: jobname AWAITS COMMAND*

Where

Table 3-3 lists the argument options and their purpose.

Table 3-3. XDM MONITOR command arguments

Argument	Purpose
<i>nn</i>	Message number to reply to.
<i>jobname</i>	End-user defined jobname running M204XDM.

Table 3-4 lists the available MONITOR commands.

Table 3-4. Available MONITOR commands

Command	Description
MONITOR MONITOR,ONLINES ONLINES	Lists all Onlines connected to an XDM master. Can be abbreviated: <ul style="list-style-type: none"> • M/MO/MON, and so on. • MONITOR,O/ON/ONL, and so on. • O/ON/ONL, and so on.
MONITOR,USERS USERS	Lists all jobs using XDM to connect to any Online. Can be abbreviated: <ul style="list-style-type: none"> • MONITOR,U/US/USE and so on. • U/US/USE and so on.
MONITOR,ONLINE= <i>jobname</i> ONLINE= <i>jobname</i>	Lists all jobs using XDM to connect to the specified Online.

The output is sent to the z/OS console and CCAPRINT in the M204XDM job. If you want another report, issue another MONITOR command. The new output is also sent to the console and is appended to the existing CCAPRINT.

Example

In the following display three job are using no XDM threads and three job are using a total of 12 XDM threads. The CAT11C job is using 4 XDM threads, and so on.

```
M204XDM.313: XDM Master Jobname=CAT141A ACTIVE (Sub-
system=CAT1)
_M204XDM.314:      Online Jobname=CAT108  Users=0
_M204XDM.314:      Online Jobname=CAT11C  Users=4
_M204XDM.314:      Online Jobname=CAT11S  Users=2
_M204XDM.314:      Online Jobname=CAT112  Users=0
_M204XDM.314:      Online Jobname=CAT109  Users=6
_M204XDM.314:      Online Jobname=CAT102  Users=0
_M204XDM.318: (Totals)  Onlines=6  Users=12
```

Using a batch job

To monitor XDM use a standalone batch job, when:

- You might not be authorized to issue console commands.
- You do not want voluminous MONITOR command output in your JES log.
- To get additional information. The batch job provides options for more detailed reporting and you can dump the control blocks for problem diagnosis.

Execute the report with the following JCL:

```
//MON      EXEC   PGM=M204XMON,PARM='parameter'
//STEPLIB DD   DSN=your.LOADLIB,DISP=SHR
//CCAPRINT DD   SYSOUT=*
```

The output goes to CCAPRINT. Unless you request otherwise, no output is written to the operator console.

The argument string can be in one of the following forms:

Syntax

```
SSNAME=ssss, ONLINE=oooooooo, DETAIL=nnn,
OUTPUT=CONSOLE
```

or

```
ssss
```

Where

Argument	Purpose
ssss	Is the z/OS subsystem name to report on (required).
oooooooo	Is the Online name to report on (optional). If not specified, reports on all Onlines using XDM.
nnn	Specifies what to report (optional). If not specified, defaults to 3.
CONSOLE	Specifies that the report is sent to both the console and CCAPRINT (otherwise to CCAPRINT only).

DETAIL=nnn is the sum of the following values:

Value	Purpose
128	Dumps XDM control blocks.
64	Interprets XDM control blocks.
2	Lists XDM Online and user details.
1	Lists XDM summary information.

If DETAIL NE 0, the return code can be one of the following values:

Value	Meaning
0	Completed normally.
4	An error was found in XDM control blocks (this is not necessarily a real error; it can occur if a control block was being modified as M204XMON ran).
8	Invalid parameter: CCAPRINT failed to open, and so on.

If DETAIL EQ 0 (not usually used), the return code can be one of the following:

Value	Meaning
0	No XDM Onlines are active.
2	XDM Onlines are active, but none have active users.
3	XDM Onlines are active, and some have active users.
4	OC4 while processing XDM control blocks; retry.

Shutting down XDM master from the console

Ordinarily you must keep XDM master active: shut it down only prior to an IPL. You can shut it down by operator command. The EOJ command, which is the normal command used to terminate XDM, checks that all connections have

been terminated and does not shut down if XDM Onlines are still active. When XDM master address space is active, it displays the following reply message on the operator console:

Syntax `*nn M204XDM.100: jobname AWAITS COMMAND`

Where

- *nn* is the message number to reply to.
- *jobname* is an end-user defined job.

Usage The following shutdown commands are available:

Command	Shuts down XDM master...
EOJ	If no Onlines are active.
EOJ,CANCEL	Even if Onlines are active.
EOJ,FORCE	Without cleaning up the cross-memory environment.

Usually you can terminate the M204XDM job via EOJ without using the FORCE option. However, when recycling the CRAM M204XDM job to apply maintenance from Rocket Software, you must follow the instruction in the Early Warning(s). There may be circumstances that require using the FORCE option or an IPL.

Warning Use EOJ,CANCEL and EOJ,FORCE commands with extreme caution. Abnormal termination of XDM master might require an IPL in order for Model 204 Online jobs to use cross-memory services or to reclaim orphaned CSA storage.

If you issue an EOJ,CANCEL or EOJ,FORCE commands, any of the following events are possible:

- Active applications abend.
- Active CRAM threads abend.
- Model 204 Online termination abends.
- CICS applications will ASRA.
- CICS might abend at shut down.

Working with CRAM

If desired, multiple subsystem names (separate IGCLM244s) can be used to isolate CRAM-XDM from the current production CRAM environment.

Restriction CRAM-XDM requires a non-swappable address space, a system LX, and is recommended as noncancelable.

CRAM OPEN processing

CRAM can operate in two modes: converser and master/user:

- Converser mode is like a telephone conversation: either side can speak and one waits for the other if there is a conflict. IFAM applications use the converser mode.
- Master/user is like half-duplex conversations: the end-user requests to speak and the Model 204 Online (master) gives permission. The Online controls the entire conversation by telling the user what to do next.

An Online makes CRAM communication possible by issuing a master CRAM OPEN command, specifying a channel name, a block size, and the number of connections (subchannels) available. The information needed to issue this OPEN is derived from parameters specified on User 0's parameter line. Subsequently, Model 204 applications can issue commands on a particular channel.

Communicating with multiple regions of Model 204

You can communicate with several regions of Model 204 concurrently by establishing a distinct CRAM channel name for each region. One IFDIAL remote User Language and several IFSTRT IFAM2 threads can communicate over a particular channel when the channel name is specified in the IFDIAL or IFSTRT call that establishes the connection. You must use one IODEV definition for each terminal.

CRAM channel names consist of 1 to 8 characters that are specified on the User 0 parameter line. Use the parameters shown in Table 3-8 on page 92 or those supplied as a default by Model 204.

CRAM channel names and parameters

Table 3-5 shows CRAM channel names, parameters, and the calls that establish each type of connection.

Table 3-5. CRAM channel names and parameters

IODEV setting	Parameter	Default name	Call to establish connection	For connection to...
11	CRFSCHNL	M204FULL		
23	IFAMCHNL	IFAMPROD	IFAM2 IFSTRT	IFAMPROD channel
			IFAM2 IFSTRTN	Channel named on IFAMCHNL
29	CRIOCHNL	M204PROD	IFAM2 IFDIAL	M204PROD channel
			IFAM2 IFDIALN	Channel named on CRIOCHNL

Required parameter settings for IODEV=29

The following parameter settings are required for IFDIAL protocol IFAM2 line-at-a-time thread (IODEV=29):

- INCCC can be set to 0 or any setting up to 32K - 4.
- INMRL must be less than or equal to LIBUFF - 4. LIBUFF can be set to any value up to 32K for the IFDIAL expanded communication buffer.
- OUTCCC can be set to 0 or any setting up to 32K - 4. It must not be larger than OUTMRL.
- OUTMRL must be less than or equal to LOBUFF - 4. LOBUFF can be set to any value up to 32K for the IFDIAL expanded communication buffer. An optional length parameter on the IFDIAL, IFWRITE, or IFREAD calls can be the standard default lengths for both input and output or an override of either default.
- First user statement (POLLNO=1) must be set with the largest CRAM buffer size (INMRL, OUTMRL). The CRAM OPEN must be performed with the largest buffer size to be used.

For general information on these parameters, see Table 3-11 on page 105.

Using CRAM on a z/OS operating system

CRAM storage requirements for z/OS

When a CRAM master opens a channel, space is allocated in the Common Service Area (CSA). You must allocate sufficient CSA space to allow the Model 204 Host Language Interface (HLI) to operate. For more information on HLI, refer to the *Model 204 Host Language Interface Reference Manual*.

The next three sections give the formula for CSA space, and explain buffer and ONLINE space allocation.

Calculating CSA space for HLI

CRAM

The following calculation determines the approximate requirements for each active HLI/Model 204 region:

$$\begin{aligned} \text{CSA Space} = & \min(nif, 1) * (68 + (nif * (IFAMBS + 100))) \\ & + \min(nul, 1) * (68 + (nul * (\max(INMRL, OUTMRL) + 100))) \\ & + \min(nfs, 1) * (68 + (nfs * (LOUTPB + 100))) \end{aligned}$$

where:

Variable	Represents
min	Minimum function.

Variable	Represents
max	Maximum function.
nif	Number of IODEV=23 users.
nul	Number of IODEV=29 users.
nfs	Number of IODEV=11 users.
IFAMBS	IFAM2 block size. IFAMBS must be set between 2 to 5 times the value of LIBUFF or LOBUFF. (Most host language parameters are assumed to be LIBUFF long and to occupy a LIBUFF portion of IFAMBS during processing by the IFAM2 interface.)
LOUTPB	Length of the output page buffer required for full-screen features.

CRAM-XDM

The following calculation determines the approximate requirements for each active XDM region:

$$\text{CSA Space} = 1510 + 28 + (50 * \text{onln}) + (30 * \text{chnl}) + (60 * \text{cnct}) + (18 * \text{asid}) + \text{trace}$$

where (the values shown are in hexadecimal):

Constant or Variable	Represents
1450	Size of the XMEMCSA module (resident in CSA)
28	SSACB length
onln	SSCB length
chnl	IICB length
cnct	UICB length)
asid	ASID table entry
trace	Trace table (length default is 3K per channel)

Note: Using CRAM-XDM, you need not calculate buffer spaces.

z/OS CRAM buffer allocation

For IODEV=29, the value of the CRAM buffer is the maximum of (INMRL, OUTMRL) + 4. If neither of these parameters is specified on the User 0 parameter line, the default of 256 bytes applies. The maximum of (INMRL, OUTMRL) + 4 determines the size of the CRAM buffer:

This parameter...	Determines the maximum length of data that...
INMRL	Can be transferred as input to Model 204.

This parameter...	Determines the maximum length of data that...
OUTMRL	Can be transferred as output from Model 204.

If an application program uses the extended communication buffer, you can set INMRL and OUTMRL up to 32K to accommodate it.

CRAM buffers are allocated as users are activated. The minimum CSA usage, without active users, is 68 bytes per channel and 100 bytes per user as defined by the number of IODEV definitions of the same type. When the first user opens a thread, the associated CRAM buffer, sized by IFAMBS, INMRL, OUTMRL, and LOUPTB parameters, is allocated.

CRAM buffers are reusable and freed only when Model 204 terminates.

z/OS ONLINE space allocation

When ONLINE is initialized, the following space is allocated:

```
CSA Space = min(nif,1) * (68 + (nif * 100))
+ min(nul,1) * (68 + (nul * 100))
+ min(nfs,1) * (68 + (nfs * 100))
```

In addition, ONLINE allocates space in the Model 204 address space based on the following formula:

```
GETMAIN space = min(nif,1) * 568
+ min(nul,1) * 568
+ min(nfs,1) * 568
+ 568 bytes allocated by each user program in its own
  address space.
```

SNAPCRAM utility for z/OS

SNAPCRAM is a program that prints out the CRAM control blocks in dump format. Sample JCL for running SNAPCRAM is:

```
//SNAPCRAM EXEC PGM=SNAPCRAM
//STEPLIB DD DSN=M204.LOADLIB,DISP=SHR
//          DD DSN=M204.CRAMLIB,DISP=SHR
//CCASNAP DD SYSOUT=A
```

If CRAM is installed in a separate library, the M204.CRAMLIB concatenation shown above is required.

Activating XDM

After you have installed Model 204 and tested it to your satisfaction, if you want to use CRAM-XDM you must add XMEMOPT=X'80' to your Online CCAIN parameters and submit the following XDM job before bringing up any Onlines, substituting with values appropriate to your site:


```
//XDMNC      EXEC PGM=M204XDM, PARM='subs', TIME=NOLIMIT
//STEPLIB DD DSN=SAM204.V630.LOADLIB, DISP=SHR
//SYSPRINT DD SYSOUT=A
//CCAPRINT DD SYSOUT=X
//
```

You must submit this job every time you IPL z/OS. TIME=NOLIMIT is required to initialize M204XDM. For 'subs', substitute CRAM-SUBSYS-NAME-VALUE from INSPARMS.

Using CRAM on a z/VSE operating system

In a z/VSE environment, CRAM lets users from other partitions communicate with Model 204 Online systems. Two versions of CRAM routines are available with each copy of Model 204:

Version...	Is compatible with...
Cross-Partition Event Control Blocks (XECB)	Single address space mode of the z/VSE operating system only.
Cross-Partition Communications Services (XPCC)	Both the single address space mode and the multiple address space mode of z/VSE.

System requirements for the XECB and XPCC versions are explained in the following sections.

Performance

To maximize performance of the CRAM subsystem, place the CRAM Load Module and both the master and user subtask programs in the System Directory List (SDL), and mark the CRAM Transient as a Move Mode transient in the SDL as well. Neither the load module nor the subtasks are SVA eligible. Any attempt to place these modules in the SVA causes errors.

System requirements (XECB version)

The CRAM subsystem uses the Cross-Partition Event Control facilities of the z/VSE operating system. To compute the number of blocks (XECBs) required for the Cross-Partition Event Control:

Syntax

$$\text{number of XECBs} = 5 * (C + N)$$

Where

Value	Represents the number of...
C	Nonmaster partitions concurrently using the facilities of CRAM.

Value	Represents the number of...
N	Master partitions concurrently using the facilities of CRAM.

These requirements affect the specification of the XECB parameter of the z/VSE Supervisor Generation Macro FOPT for releases of z/VSE prior to Release 1.3.0. Refer to the IBM *DOS/z/VSE System Generation Manual* for a further discussion of Cross-Partition Event Control Blocks.

Partition requirements

The entire CRAM subsystem runs in the partition GETVIS area of each partition that uses the facilities of CRAM:

- Each master partition has a copy of the phase IGCLM244 and the master subtask CRAMZWT.
- Each nonmaster partition has a copy of the phase IGCLM244 and the nonmaster subtask CRAMSWT.
- If a partition is both a master and a nonmaster, it has only one copy of the phase and one copy of the master and nonmaster subtasks.

Note: The storage requirements given do not include buffer storage (one I/O buffer per thread). Buffer sizes are:

```

MAX (INMRL,OUTMRL)  for IODEV=29
IFAMBS              for IODEV=23
LFSCB               for IODEV=11

```

Calculating partition requirements

Total storage requirements consist of fixed and variable storage.

Fixed storage requirements are given in Table 3-6.

Table 3-6. Fixed storage requirements

Fixed storage area	XECB version (bytes)	XPCC version (bytes)
CRAM phase	2560	3072
CRAM master subtask	13264	22144
CRAM nonmaster	13264	19840
TOTAL	29088	45056

Variable storage requirements, represented schematically, are given in Table 3-7.

Table 3-7. Variable storage requirements

Variable storage area	Bytes
Subtask communications area	A
Master channel storage	B
User (nonmaster) channel storage	C
TOTAL	A + B + C

Sizing the subtask communications area

One subtask communications area is required for each subtask running in the partition. If one program is both a master and nonmaster CRAM user, then both subtasks are present in the partition.

In the XECB version

The size of each communications area (A) is:

$$A = n * 128$$

where:

Variable	Is calculated as...
n	$(652 + (112 * (nparts - 1))) / 128$, rounded up to the next integer
nparts	Number of partitions in the z/VSE system. Round all totals up to the next higher integer.

For example, if *nparts* in the z/VSE system is 8, the computation is:

$$n = (652 + (112 * (8 - 1))) / 128 \text{ rounded up to next integer}$$

$$n = (11.2) = 12$$

$$A = 12 * 128 = 1536 = \text{Subtask Communication Area}$$

In the XPCC version

The size of each communications area (A) is:

$$A = m * 768$$

where *m* is the number of subtask communications areas. *m* has the value of 1 for each subtask. For example:

If the program is...	Value of <i>m</i> is...
CRAM master or a CRAM nonmaster	1
Both a CRAM master and a CRAM nonmaster	2

Sizing channel storage**In the XECB version**

For each master channel that is opened by a program, an Internal Interregional Control Block (IICB) is created. To compute the amount of storage required for each IICB (I):

$$I = i * 128$$

where:

$$i = (32 + (40 * \text{number of threads})) / 128, \text{ rounded up to next integer}$$

For example, if the number of master channels being opened is 3, and the channels have 4, 5, and 2 threads, respectively, the computation is:

Channel 1

$$i = ((32 + (40 * 4)) / 128) \text{ rounded up to next integer}$$

$$i = \text{rounded up } (1.5) = 2$$

$$I = 2 * 128 = 256$$

Channel 2

$$i = ((32 + (40 * 5)) / 128) \text{ rounded up to next integer}$$

$$i = \text{rounded up } (1.8) = 2$$

$$I = 2 * 128 = 256$$

Channel 3

$$i = ((32 + (40 * 2)) / 128) \text{ rounded up to next integer}$$

$$i = \text{rounded up } (0.8) = 1$$

$$I = 1 * 128 = 128$$

The sum of the values of I (640 bytes) calculated for each channel is the total channel storage requirement.

In the XPCC version

For each channel opened in the master environment, the storage requirement is:

$$B = (p + t + 1) * 128 \text{ bytes}$$

where:

Variable	Is calculated as...
p	(160 + 40 * t)/128 rounded up to the next highest integer
t	Number of threads in the channel

The storage requirement for each nonmaster thread is 256 bytes. The total storage requirement is computed by:

$$C = 256 * t \text{ bytes}$$

SNAPCRAM utility

The following JCL statements are required to execute the SNAPCRAM utility:

```
//JOB SNAPCRAM
//DLBL M204LIB, 'M204.PROD.LIBRARY'
//EXTENT SYSnnn, ...
//LIBDEF PHASE, SEARCH=M204LIB.V411
//EXEC SNAPCRAM, SIZE=AUTO
/&
```

SNAPCRAM interacts with the operator in the following way:

1. As part of its initialization process, the utility asks for a command with an ENTER COMMAND prompt on the console. Enter one of the following SET commands:

Syntax XECB version

```
SET COUNT=nnn, INTERVAL=iii
```

Syntax XPCC version

```
SET NAME=channel, COUNT=nnn, INTERVAL=iii
```

Where

Variable	Is
<i>channel</i>	One of the active ONLINE system channel names for which a SNAP dump is requested.
<i>nnn</i>	Number of dumps requested. The default is 1 dump.
<i>iii</i>	Time interval between dumps (in seconds). The default is 5 seconds.

NAME, COUNT, and INTERVAL can be abbreviated to N, C, and I.

SET is useful for tracing CRAM control blocks when CRAM is not in a soft wait state.

RUN is issued after SET and begins the dump.

STOP terminates the SNAPCRAM operation.

2. After the specified number of dumps has been produced, SNAPCRAM redisplayes ENTER COMMAND.

To interrupt the dump, issue the z/VSE MSG *pp* command, where *pp* is the SYSLOG ID of the partition in which SNAPCRAM is running. The ENTER COMMAND prompt is displayed after the interruption.

CRAM IFSTRT protocol IFAM2 (IODEV=23) requirements

IFAM2 is a multithread configuration of Model 204 that supports host language calls to the Host Language Interface (HLI) from one or more user programs. The user programs normally run as separate jobs in other regions or partitions and share a single copy of the database management system software. Each user program must have a small interface module linked in its region to enable communication between the user program and the IFAM2 region through a special interregional supervisor call.

The concept of a Host Language Interface thread corresponds to that of an Online user for ONLINE. As with an Online user, each Host Language Interface thread is defined by a user's parameter line in the CCAIN input stream.

In a z/OS environment

- Communication between IFAM2 and user-written host language programs requires CRAM.
- Support for IFAM2 and CICS is provided. Host language applications can run in 31-bit addressing mode.

In a z/VSE environment

- IFAM2 requires CRAM for the transfer of data and commands between the ONLINE configuration in another z/VSE partition and the user program.
The partition in which IFAM2 application programs are run must have access to a core image library containing the modules IGCLM244 and CRAMSWT.
- You must initialize the ONLINE program with one IODEV=23 for each concurrent IFAM2 thread in use.
- For database files accessed by the IFAM2 program, you must provide label information in the JCL that executes the ONLINE program with which the IFAM2 program is communicating.

Multiple Online copies on a single system

When running multiple Onlines concurrently, you access a specific Online by defining an alternate channel name for that Online and specifying the name in an IFAM CALL. Concurrently operating Onlines require different names for their CRAM channels. If duplicate channel names are used, the second version's job is terminated with a job step return code of 96. Table 3-8 shows the default channel name and how to define an alternate channel name.

Table 3-8. Default and alternate channel names

IODEV=	Access method	Default channel name	Defines an alternate channel name with...
11	Full-screen	M204FULL	CRFSCHNL parameter on the User 0 parameter line.

Table 3-8. Default and alternate channel names (Continued)

IODEV=	Access method	Default channel name	Defines an alternate channel name with...
23	IFSTRT protocol IFAM2 threads	IFAMPROD For CICS transactions: M204	IFAMCHNL parameter on the User 0 parameter line. To access the Model 204 version, call IFSTRTN and specify the alternate channel name. For more information about IFSTRTN, see the <i>Rocket Model 204 Host Language Interface Reference Manual</i> .
29	IFDIAL protocol IFAM2, line-at-a-time	M204PROD	CRIOCHNL parameter on the User 0 parameter line. To access the Model 204 version, call IFDIALN and specify the alternate channel name. For more information about IFDIALN, see the <i>Rocket Model 204 Host Language Interface Reference Manual</i> .

TSO Interface

For the TSO Interface, specify an alternate channel name when the terminal connection is made to Model 204:

- If TSO is installed as a command processor (CP), invoke Model 204 as follows:

```
M204FS [subsystem:] channel-name (for IODEV=11)
M204TTY [subsystem:] channel-name (for IODEV=29)
M204PROD [subsystem:] channel-name (for IODEV=29)
M204FULL [subsystem:] channel-name (for IODEV=11)
```

M204FULL is the default.

- If TSO is not installed as a CP, invoke Model 204 as follows:

```
[M204FS]
CALL 'library-name' [(M204TTY) | (M204FULL) | (M204PROD)]
    '[subsystem:] channel-name'
```

where

Default channel name for...	Uses this terminal connection...
M204FULL	M204FS
M204PROD	M204TTY

INTERCOMM Interface

For the INTERCOMM Interface, specify an alternate channel name when the terminal connection is made to Model 204 as follows:

```
verb,channel-name
```

CICS Interface

With the CICS Interface, invoke Model 204 as follows:

```
M204 [subsystem:]channel-name (for IODEV=11)
L204 [subsystem:]channel-name (for IODEV=29)
```

SQL server threads (IODEV=19)

Model 204 requires a pool of threads to support server control of Horizon SQL connections. The system manager must define these threads in the CCAIN stream as part of the Online startup configuration.

IODEV	Defines thread for...	You must define this thread for...
19	Model 204 SQL Horizon	All Connect★ configurations.

Horizon code is incorporated directly into the core of Model 204. You have the opportunity to try Horizon and Connect★ without additional expense. The number of threads is limited to two. If you want additional threads, please notify Technical Support.

For complete information on Model 204 SQL system management, including SQL catalog and procedure file maintenance, SQL parameters and statistics, and processgroup definitions, see the *Rocket Model 204 SQL Server User's Guide*.

Coding the IODEV=19 line

Defining the IODEV=19 thread requires specifying or changing the following parameters. For an example of the specification of an IODEV=19 line, see “Example for z/VM” on page 95.

Note: Define these as the last IODEV in your CCAIN as some of the parameters could have an affect on other IODEV threads.

NOTHREAD

You can insert several IODEV=19 lines in the CCAIN stream. On the first line defining the SQL thread, include the NOTHREAD parameter to indicate the number of SQL threads to be allocated. Set NOTHREAD to the number of concurrent Connect★ conversations to be supported.

SQLBUFSZ

SQLBUFSZ is a required resettable User 0 parameter that corresponds to the size of the Model 204 SQL Server buffer that assembles incoming SQL messages from a receiving buffer (in CRAM, IUCV, or Horizon). The SQLBUFSZ value defines the maximum length of an incoming SQL message.

Because the largest incoming SQL message is likely to be a DDL transaction, set SQLBUFSZ large enough to accommodate your largest DDL transaction plus 50 bytes of overhead. You can also monitor the Model 204 SQLI statistic to determine the size of the largest SQL input request.

The recommended initial value for this parameter is 100000. Specify it on the first IODEV=19 line.

INMRL

INMRL is a user parameter that determines the maximum input line length for an I/O device as well as the maximum length of an internal Model 204 buffer. You must set INMRL to a value greater than or equal to 500 (characters) for z/OS IODEV 19 threads. Specify it on the first IODEV=19 line.

Processing complex SQL statements might require a larger INMRL setting. If the INMRL buffer capacity is exceeded, you get the Model 204 error message:

```
INVALID STRING ARGUMENT
```

Increase INMRL by 50% and rerun your request.

NSUBTKS

Increase the User 0 NSUBTKS parameter by 3 per open Horizon link for Connect★ processing.

SQLIQBSZ

The user parameter SQLIQBSZ determines the size in bytes of the internal buffer used to compile and evaluate SQL requests.

The minimum value is 2032; the maximum is 32752. The maximum setting (large enough for a 250-column table with an average column length of 30 bytes) is sufficient for most SQL applications.

For complete details on SQLIQBSZ, see the *Rocket Model 204 Parameter and Command Reference*.

Example for z/VM

The following example shows the CCAIN data stream for a z/VM job that brings up a Model 204 Online for SQL processing. This example shows sample settings of some of the CCAIN parameters described earlier in this chapter, the

first two IODEV lines for each SQL IODEV, and Model 204 SQL DEFINE commands.

```
••• LIBUFF=3000,LNTBL=600,LOBUFF=5000,LPDLST=32760,
LQTBL=2000,LSTBL=12000,LTTL=150,LVTBL=300,
MINBUF=18,MAXBUF=200,NSUBTKS=15,•••
SERVSIZE=700000,•••
•
•
•
IODEV=49,POLLNO=1,NOTHREAD=4
IODEV=49,POLLNO=2
IODEV=49,POLLNO=3
IODEV=49,POLLNO=4
IODEV=19,POLLNO=1,NOTHREAD=4,SQLBUFSZ=100000,LHEAP=200000, -
LIBUFF=6048,SQLIQBSZ=32752
IODEV=19,POLLNO=2
IODEV=19,POLLNO=3
IODEV=19,POLLNO=4
* above IODEV=49 is for RCL connections
* above IODEV=19 is for SQL CONNECT* connections
*-----* /
*  DEFINE CONNECT * SQL AND RCL LINK ETC...      * /
*-----* /
*****
DEFINE LINK TCPSQL WITH SCOPE=SYSTEM TRANSPORT=TCPSE -
  PROTOCOL=IP LOCALID=ANY INBUFSIZE=4096 CONNECTIONS=8 -
  SERVPORT=2132
DEFINE PROCESSGROUP ANY192  WITH SCOPE=SYSTEM LINK=TCPSQL -
  INLIMIT=8 OUTLIMIT=8 REMOTEID=192.0.0.0  LOGIN=NOTRUST  -
  GUESTUSER=REJECT  MASK=255.0.0.0
DEFINE PROCESSGROUP ANY204  WITH SCOPE=SYSTEM LINK=TCPSQL -
  INLIMIT=8 OUTLIMIT=8 REMOTEID=204.0.0.0  LOGIN=NOTRUST  -
  GUESTUSER=REJECT  MASK=255.0.0.0
DEFINE PROCESS CCARSQL WITH SCOPE = SYSTEM -
DATALEN = 32763 FROM = (ANY192, ANY204)
*****
```

Horizon (IODEV=27)

Horizon, the Model 204 distributed application facility, enables Model 204 applications to participate in program-to-program processing by communicating through SNA Communications Server with one or more other programs. The applications communicate over SNA LU 6.2 sessions (with a partner application that can run in the same or in a different computer) in a logical connection called a conversation. Support for such conversations is configured primarily by Model 204 DEFINE commands and is fully described in the *Model 204 Horizon: Intersystem Processing Guide*.

Horizon code is incorporated directly into the core of Model 204 by default. You have the opportunity to try Horizon without additional expense. The number of threads is limited to two. If you want additional threads, contact Rocket Software.

Horizon threads are also used for SQL server processing (IODEV=19). For information on IODEV=19, see “Coding the IODEV=19 line” on page 94.

Managing IODEV=27 threads

In CCAIN, the system manager needs to include only IODEV=27 statements. For LU 6.2 sessions, all but the first of these statement lines have no further parameters. The first statement line includes the NOTERM parameter, which indicates the total number of IODEV=27 threads to be in the system. For IODEV=27, you can use the synonym NOTHREAD in place of NOTERM.

In a Horizon conversation, an application program either initiates the conversation (client) or is started up by a request for a conversation from a partner application (server). IODEV=27 statements are required for a Model 204 ONLINE only if it has server applications that can be started up by clients in the network. The number of IODEV=27 lines determines the maximum number of server applications that the ONLINE can allow to run concurrently. This number has no effect on the number of concurrent conversations that client applications on the local system can initiate on other remote systems.

IODEV=27 threads are thus needed only on one end of a Horizon conversation: the server end. Since the client end is started up by a request from a local user and not by a request from a remote program, the client is part of the local user thread. As such, the client is supported by whatever IODEV statement was used to set up that user thread.

Support for the additional SNA Communications Server APPL required with Horizon is configured in DEFINE commands rather than in CCAIN, as explained in the *Model 204 Horizon: Intersystem Processing Guide*.

IUCV (IODEV=39, 41, 43)

A Model 204 Online can be run as multiuser or single-user (batch) operation, or single-user within a user's z/VM virtual machine. BATCH204 is not supported, but can be simulated by executing the ONLINE module in a CMSBATCH machine or in a user-defined virtual machine.

Communication between an individual z/VM user machine and the Model 204 service machine running in a separate z/VM virtual machine is provided through IUCV. The access path between a CMS terminal user and Model 204 through IUCV is established by means of a channel that is defined on User 0's parameter line. Each channel name must be unique within the Model 204 service machine. If no name is specified, defaults are used.

For example, the following statement connects a user to a Model 204 service machine:

```
M204 USER machineid CHAN M204VMFS
```

Where

- *machineid* is the name of the service machine and must be specified.
- *M204VMFS* is the channel name.

The section “M204 syntax” on page 101 describes the M204 command in detail.

Table 3-9 lists the parameters used to specify channel names.

Table 3-9. Parameters for IUCV channel names

IODEV setting	Parameter	Default name	You can name an alternate channel on the...
39	VMIOCHNL	M204VMIO	VMIOCHNL parameter.
41	VMFSCHNL	M204VMFS	VMFSCHNL parameter. Used in partner process communication and in communication with 3270 and 3270-compatible local and remote terminals.
43	VMIFCHNL	M204VMIF	VMIFCHNL parameter.

Programs required for the IUCV CMS Interface

The programs required to support the IUCV CMS Interface in communicating with ONLINE configurations operating in a virtual machine are distributed on a CMS format tape. Table 3-10 lists the required files. Information about installation is in the *Model 204 z/VM Installation Guide*.

Table 3-10. CMS files required for the IUCV Interface

File name	File type	Function
M204	EXEC	Initiates communication between the user and Model 204
M204	HELPCMS	Provides HELP information for M204 command syntax
M204IFAM	EXEC	Aids in establishing an HLI connection to Model 204
M204IFAM	TXTLIB	Specifies the library of object modules that must be included in IFAM2 programs under CMS
M204INFO	MODULE	Provides information about the CMS environment
M204USR	LOADLIB	Provides interactive access to Model 204 and parses the command string, but used for nucleus extension

Table 3-10. CMS files required for the IUCV Interface

File name	File type	Function
M204USR	MODULE	Provides interactive access to Model 204 and parses the command string

Interactive access (M204USR MODULE)

Interactive access to Model 204 from CMS, which is provided by the M204USR MODULE file, permits a user at a terminal to communicate with Model 204 executing on another virtual machine under CMS.

The M204USR MODULE file executes in the CMS user area in the following manner:

1. Parses a command string that defines the parameters of a communication interface between CMS and Model 204.
2. Uses these parameters to establish the connection between the user's virtual machine and Model 204.
3. Accepts data for display and reads input from the terminal for transmission to Model 204 through IUCV.

M204USR MODULE provides the IUCV interface that allows a CMS user to communicate with a Model 204 Online running in a CMS service machine. You can run the interface in either the user area, a discontinuous saved segment (DCSS), or a nucleus extension. The TPROCESS communication facility requires IUCV to be run as a saved segment or a nucleus extension. Saved Segments are discussed in the *Model 204 z/VM Installation Guide*.

The M204USR MODULE and a load library (M204USR LOADLIB) for the user-area and nucleus extension versions are built by the M204GEN EXEC. The DCSS version (M204USR) is built by M204XGEN EXEC. Keyword options specified in the distributed M204 EXEC can execute in any of the three nodes. If no options are specified, the default is DCSS.

Line-at-a-time (IODEV=39)

The IUCV line mode thread is used to communicate with any terminal supported by z/VM. Determine the buffer size by the value of MAX(INMRL,OUTMRL) minus 12 bytes. (See Table 3-11 on page 105 for a summary of user parameters.)

- INMRL must be less than or equal to LIBUFF minus 4. You can set LIBUFF to any value up to 32K for the IFDIAL expanded communication buffer.
- OUTMRL must be less than or equal to LOBUFF. You can set LOBUFF to any value up to 32K for IFDIAL expanded communication buffer.

The maximum of (INMRL, OUTMRL) plus 4 determines the size of the CRAM buffer. INMRL determines the maximum length of data that can be

transferred for input to Model 204. OUTMRL determines the maximum length of data that can be transferred as output from Model 204.

An optional length parameter on the IFDIAL, IFWRITE, or IFREAD calls can be the standard default lengths for both input and output or an override of either default.

- You can set OUTCCC to 0 or any setting up to 32K minus 4. It must not be larger than OUTMRL.
- You can set INCCC to 0 or any setting up to 32K minus 4.

IUCV full-screen (IODEV=41)

The full-screen mode thread is used to communicate with IBM 3270 and compatible local and remote display terminals. This thread is also used in partner process communication.

The buffer size is 12 less than the value of the LOUPTB parameter. For full-screen I/O, set the LOUPTB parameter to at least 24044.

IFAM2 (IODEV=43)

In a CMS environment, Model 204, when executing in the service virtual machine, communicates with host language programs that run in one or more CMS virtual machines.

When using IFAM2:

- You can use z/VSE macros within IFAM2 programs.
- You can compile IFAM2 programs with a z/VSE compiler.
- IFAM2 provides multithread access to Model 204 files from a host language program.
- You define the Model 204 files accessed by the host language programs to the service machine just as you define other database files. Define application program files (if any) in the user's CMS virtual machine.

IUCV is not supported in the VAE mode of z/VSE operating systems.

RCL thread (IODEV=49)

Model 204 requires a pool of Horizon threads to support server control of RCL connections. The system manager must define these threads in the CCAIN stream as part of the Online startup configuration.

IODEV=49 defines a Model 204 Remote Command Line (RCL) Horizon thread. The communication software calls a server module, which recognizes the command data coming to an IODEV=49 input thread and passes it to the routine that processes all Model 204 commands. Each RCL thread requires an IODEV=49 statement in the CCAIN stream.

Using an IODEV=49 thread, all output generated on the mainframe Model 204 server by the executing commands is formatted as if it were retrieved from an SQL table composed of a single column of varying character data. Thus, the client application can retrieve the data using SQLFetch commands.

The following example defines two IODEV=49 threads:

```
IODEV=49, POLLNO=1, NOTHREAD=2
IODEV=49, POLLNO=2
```

NOTHREAD parameter

You can insert a number of IODEV=49 lines in the CCAIN stream. On the first line defining the RCL thread, include the NOTHREAD parameter to indicate the number of RCL threads to be allocated. Set NOTHREAD to the number of supported concurrent Connect★ RCL connections. The maximum must equal the number of concurrent Connect★ users specified in your purchase agreement for the product.

M204 command (CMS)

M204 syntax

The M204 command establishes a connection with the Model 204 database management system, which allows logon and use of Model 204.

The format of the M204 command is:

Syntax

```
M204 [options ... [ ] ]

[LINE | DISplay | DCSS | NUCEXT | UAREA] [USERid userid]

[CHANnel channel] [LOGin | NOLogin] [DISConn string]

[SUBSet string] [CMD filename] [ONLINE | IFDIAL]
```

Where

- *LINE* specifies a line mode connection. LINE is the default for all terminals except 3270 display terminals.
- *DISPLAY* specifies a full-screen mode display. DISPLAY is the default for 3270 terminals.
- If DISPLAY is selected as the default and the CHANNEL parameter is not specified, an unsuccessful attempt to obtain a connection causes an automatic attempt to connect in LINE mode.
- *DCSS* causes the EXEC to load the saved segment version of the Model 204 IUCV Interface program (M204USR). This is the default.
- *NUCEXT* causes the EXEC to load the Model 204 IUCV Interface program as a nucleus extension.

- *UAREA* causes the EXEC to run the Model 204 IUCV Interface program by running a module in the user area.
- *USERID* specifies the user identifier of the virtual machine in which the Model 204 database management system is executing.
An installation-dependent default value is provided for the USERID parameter if it is not specified explicitly. If the parameter value is specified as an asterisk (*), Model 204 is invoked in single-user mode on the user's virtual machine.
- *CHANNEL* specifies the Model 204 channel connection. Installation-dependent defaults are provided for LINE and DISPLAY mode connections if an explicit CHANNEL parameter value is not specified.
The CHANNEL option is ignored if Model 204 is invoked in single-user mode.
- *LOGIN* specifies an automatically generated initial LOGIN command. The LOGIN option is ignored if Model 204 is invoked in single-user mode.
- *NOLOGIN* specifies no automatically generated initial LOGIN command. The NOLOGIN option is ignored if Model 204 is invoked in single-user mode.
- *DISCONN string* specifies the character string recognized as the Model 204 disconnect sequence.
DISCONN causes transmission of a DISCONNECT command to Model 204 if the designated disconnect string is identified as the only input data on the screen. DISCONN is ignored if Model 204 is invoked in single-user mode.
- *SUBSET string* specifies the character string recognized as the CMS SUBSET escape sequence.
SUBSET causes CMS SUBSET to be entered if the designated escape sequence is identified as the only input data on the screen. SUBSET is ignored if Model 204 is invoked in single-user mode.
- *CMD filename* specifies the CMS file used to supply input data for Model204. (See "Command File facility" that follows.)
Records in the file are read one at a time and used as input in response to requests from Model 204. If the end of the file is reached before the connection with Model 204 is broken, subsequent input is requested from the terminal.
- *ONLINE* specifies, for single user invocation (USER *), a normal Online connection type. A default setup EXEC named SINGLUSR EXEC is assumed. ONLINE is the default connection type.
- *IFDIAL* specifies, for single-user invocation (USER *), to run an IFDIAL connection type. A default setup EXEC named SINGDIAL EXEC is assumed.

Command File facility

The initial entry to a Model 204 application from CMS can be preprogrammed by using the CMD option of the M204 command to specify a file of commands as the initial source of input to Model 204 from CMS. The command file can include applications that combine the use of Model 204 with other CMS-based facilities.

Only a single command file can be used to provide input at the beginning of a Model 204 session. The following limitations apply:

- Only the first field can be filled in on a full-screen panel.
- Program function and other interrupt keys cannot be signaled. Pressing the Enter key is simulated for each line.
- Conditional execution of commands, interpretation of screen contents, or determination of the effects of input are not provided.

A command file is implemented by defining the file with the keyword CMD, followed by a 1- to 8-character file name. The command file can reside on any accessed disk in the CMS user's virtual machine. A file type of M204CMND is required for the file specified in the CMD option.

For example, the following statement causes CMS to read the file name with a file type of M204CMND and pass the file contents to Model 204:

```
M204 USER userid CMD filename
```

The following statements are typical of a command file:

```
LOGIN
OPEN TEST
DICTIONARY
```

Where

- Login password prompts are bypassed.
- *OPEN TEST* does not require a password.
- *DICTIONARY* is the application automatically presented to the user.

Command file processing

The command file can be defined as either fixed or variable format with a record length of up to 256 characters. Variable format is recommended.

If a full-screen thread is used, each line of the file is treated as a single full-screen data stream. Each line is presented to Model 204 as if the Enter key were pressed immediately after the data.

If a line-mode thread is used, each line acts as the response to a terminal read request from Model 204. Each line is presented to Model 204 as if the Return key were pressed immediately after the data.

Automatic logon

To avoid including Model 204 passwords in the command file and compromising the security of the Model 204 system, the use of the automatic logon feature for IUCV threads is recommended:

- Automatic logon can be made the default for IUCV threads by modifying the default section of the distribution tape version of M204 EXEC from &LOGIN to &LOGIN=LOGIN.
 - Automatically generated login defaults to the CMS user ID invoking the M204 EXEC when no other account is specified. Entering the login with a new ID overrides the default user ID.
 - Account name default in the initial login command can be overridden if the M204 EXEC is invoked without automatically generating the initial login command (see below) and the LOGIN/LOGON command is issued with the user ID.
 - If the SYSOPT parameter is not set to require logins, Model 204 grants superuser privileges even though the login failed.
- Automatic logon cannot be generated for a single-user version of Model 204. However, the following CCAIN input stream provides a substitute:

```
PAGESZ=6184,NUSERS=1,NSERVS=1
LOGIN
```

The M204 EXEC default automatic login can be overridden with:

```
M204 USERid userid CHANnel channel NOLogin
```

The M204 EXEC default manual login can be overridden with:

```
M204 USERid userid CHANnel channel LOGIN
```

Bypassing password prompts

Use the LOGCTL command with the NP option to bypass password prompts:

```
LOGCTL NP CMS
```

LOGCTL NP CMS remains in effect until the command is reversed with:

```
LOGCTL P CMS
```

To bypass password prompts, the following conditions are required:

- LOGCTL NP CMS command must be issued.
- User ID on the LOGIN/LOGON command must equal the CMS user ID of the CMS virtual machine on which the login originated.

If the default user ID is not identical to the CMS user ID, Model 204 then prompts for a password, and continues login processing.

- User ID must be located in the password table.

If the user ID is not in the password table, Model 204 prompts for a password and then rejects the login command.

CMS service machine console (ALTIODEV=45, 47)

ALTIODEV, applicable only to CMS single-user mode, is the IODEV number for continuing User 0's input. After end-of-file is reached in the User 0 file, the input for User 0 is directed to the specified device.

Setting the ALTIODEV parameter

To set the ALTIODEV parameter, place it on the program stack in the single-user EXEC rather than defining it in the CCAIN file. All other FILEDEF and CCAIN parameters are the same as those for multiple-user execution, except that no other users are defined and no IODEVs are coded.

Executing in single-user mode

To execute in single-user mode:

1. Customize the single-user EXEC and single-user CCAIN (see "CMS CCAIN file" on page 17).
2. Enter:

```
M204 USERID *
```

User environment control parameters

Table 3-11 lists parameters that you can set for individual users, either on User 0's parameter line or on the user's parameter line. Not all parameters can be reset.

For a complete description and recommendation for each parameter, see:

- *Model 204 Parameter and Command Reference* and the *Model 204 Terminal User's Guide*
- If you do SQL processing, see also the *SQL Connectivity Guide*

Table 3-11. User environment control parameters

Parameter	Specifies...
AUTOSYS	Automatic application subsystem invoked upon login. The default is a null string.
CAUDIT	Type of auditing of input lines for the production of CI, CS, or CP journal lines. The default is 0.

Table 3-11. User environment control parameters (Continued)

Parameter	Specifies...
EDIT	Controls the type of editing performed on input lines. The default is 0.
FSATTN	Key interpreted as an attention key of 3270 full-screen terminals. The default is PA1 (X'6C').
FSTRMOPT	Full-screen terminal options. The default is 0.
HDRCTL	Page header formatting. The default is 0.
HTLEN	Maximum length of a User Language header or trailer. The default is 132.
INCCC	Input continuation character column. When data is input, any nonblank character appearing in column number INCCC indicates that input is continued on the following line. The value of MODEL determines the default value of INCCC. A hyphen can be used to continue long input lines within User Language requests and some commands.
INMRL	Maximum input line length. The value of MODEL determines the default value of INMRL. The recommended value for SQL processing is at least 500.
INPUT	Name of the input file on a device type (BSAM, or teletype) receiving input. The INPUT name must match the names specified on FILEDEF or JCL DD statements.
IODEV	Input/output device type.
LAUDIT	Logical input line audit. The default is 1.
LCPDLST	Dynamically allocated space for the C pattern-matcher. The default is 2176. For SQL processing, use the default.
LECHO	Echoing of user input; set it to 0 for Online users. The default is 1, which copies every line from the input device to the output device. For Online users (unlike User 0) these devices are the same.
LFSCB	Length of the full-screen buffer. The default is 0.
LFTBL	Length of the file group table (FTBL). The default is 1000. Adjustment to the FTBL size can be made with the UTABLE command while files are open, including when the user is in a subsystem.
LGTBL	Length of the global variable table (GTBL). The default is 288, which is required for a subsystem.

Table 3-11. User environment control parameters (Continued)

Parameter	Specifies...
LIBUFF	<p>Length of the input buffer used for input lines from CCAIN or from a user's terminal. LIBUFF must be three bytes longer than the longest line or record read into it. Longer lines are rejected with an error message. The default is 250.</p> <p>If an input line is continued with a nonblank character in column INCCC, the number of characters in the input line (the original line and all continuations, not including the continuation characters) is limited to the value of LIBUFF.</p> <p>The value of LIBUFF is usually much higher for IFAM threads than for terminal users.</p> <p>For SQL processing, the recommended value is 10000.</p>
LITBL	Length of the dummy string and \$READ response table (ITBL). The default is 0.
LNTBL	Number of 12-byte entries in NTBL. The default is 50.
LOBUFF	Length of the output buffer used for output lines to CCAAUDIT, CCAJRN, CCAPRINT, a user's terminal, or a directed output (USE) data set. The value of LOBUFF must be greater than or equal to the value of OUTMRL. The recommended value for SQL processing is 5000. The default is 256.
LOGONENQ	Use of a unique user ID for logon to a single ONLINE system from specific terminals. LOGONENQ cannot be reset. The default is 0. All nonunique user ID terminals must be listed before unique user ID terminals. Specification of LOGONENQ on an IODEV specification affects all subsequent IODEV definitions.
LOUTPB	<p>Length of the output page buffer for 3270 and 3275 display stations. The minimum is determined by the value of MODEL.</p> <p>A nonzero value, not greater than the page size, must be specified for terminals supporting active backpaging or full screen usage. 3000 is recommended for full screen use. IODEV=11, used with the pseudo conversational interface, requires a value slightly larger than the screen size. Model 5 terminals may require over 3000. 2130 is suggested for other uses.</p>
LPDLST	Length of the user pushdown list. The default is 2600. The recommended initial value for SQL processing is 32K.
LQTBL	Number of 16-byte entries in QTBL. The default is 400. LQTBL may need to be increased approximately 1% for User Language requests to accommodate quad sum expansion.
LSTBL	Length of the character string table (STBL). The default is 600.
LTTBL	Number of 4-byte entries in the translation table (TTBL). The default is 50. For SQL processing, the recommended initial value is 2000.

Table 3-11. User environment control parameters (Continued)

Parameter	Specifies...
LVTBL	Number of 32-byte entries in the compiler variable table (VTBL). The default is 50.
LVLTRC	Level of User Language INCLUDE statements reached when the given input stream is processed. Used as a diagnostic tool. The default is 0 (off).
LXTBL	Length of the security information table (XTBL). The default is 1000. Adjustment to the XTBL size can be made with the UTABLE command while files are open, including when the user is in a subsystem.
MAXHDR	Maximum number of output page header lines that can be defined within a single User Language request. The default is 5.
MAXTRL	Maximum number of output page trailer lines that can be defined within a single User Language request. The default is 5.
MODEL	3270 terminal model number. Establishes an appropriate screen size by setting values for: <ul style="list-style-type: none"> • INCCC • INMRL • OUTCCC • OUTLPP • OUTMRL The default is 2.
NBKPG	Maximum number of output backpages that can be retrieved and displayed with the backpage command. The default is 0.
NORQS	Maximum number of temporary procedures saved for each user. The default is 5.
NOTERM	Number of terminals defined with the IODEV parameter as a group on user parameter lines. The default is 0. NOTERM is used in conjunction with CMS POLLNO terminals or remote User Language threads. NOTERM must be specified on the first parameter line of the sequence (the POLLNO=1 line). There are no physical lines and only one logical line for SNA Communications Server terminals and remote User Language threads.
NOUTBUF	Number of full-screen output buffers allocated and used by the Model 204 SNA Communications Server 3270 Interface.
NSUBTKS	Maximum number of pseudo subtasks that can be generated during a run. Specification of the number of subtasks applies to SNA Communications Server.

Table 3-11. User environment control parameters (Continued)

Parameter	Specifies...
OUTCCC	Size of an output line. The default is 132 unless the MODEL parameter has been set. In that case, MODEL determines OUTCCC's default. OUTCCC is normally set to the same value as OUTMRL. A value greater than OUTMRL causes long output lines to be truncated at OUTMRL characters. If an output line is longer than OUTMRL, it is broken into sections containing up to (OUTCCC - 1) characters on each line.
OUTLPP	Number of lines per page, including input, output, page headers, and trailers. The default is 56 unless the MODEL parameter is set. In that case, MODEL determines the default. When OUTLPP=0, output comes in a steady stream with no headers or trailers.
OUTMRL	Maximum output line length. The default is 132 unless the MODEL parameter is set. Lines are shorter than OUTCCC if OUTCCC is greater than 0 and less than or equal to OUTMRL. In that case, MODEL determines OUTMRL's default.
OUTPUT	Name of the output file on a device type (BSAM, or teletype) receiving output. The OUTPUT name must match the names specified on FILEDEFs or JCL DD statements.
POLLNO	Relative position of terminals in the Online user queue. Arrange POLLNO values consecutively from 1 to the highest number. The default is 0.
PRIORITY	User's priority class. Settings are LOW, STANDARD, and HIGH. Priority decisions are made on the basis of internal priority settings and ranges. (See the discussion of the PRIORITY parameter and command in "Priority scheduling" on page 135.) The default is STANDARD.
RETRVKEY	PF key used to retrieve a previous terminal input line. 280 bytes of spare core are required for each user that has a defined retrieve key. The default is 0.
SERVNSA	(server non-swappable areas) A bit setting indicating the tables that you want to allocate in above the bar storage. At this time, only FTBL may be selected and its appropriate setting is x'02000000'.
SERVNSSZ	(server non-swappable size) The amount of space in bytes required for the above the bar server tables per user. SERVNSSZ is a user 0 parameter applicable to all users. The total amount of storage allocated for non-swappable server parts equals SERVNSSZ rounded to 4 K and multiplied by NUSERS. When sizing SERVNSSZ, use the largest FTBL sizes that might be needed.

Table 3-11. User environment control parameters (Continued)

Parameter	Specifies...
SQLBUFSZ	Size of the Model 204 SQL Server buffer that assembles incoming SQL messages from a Horizon, CRAM SQL, or IUCV SQL buffer. The recommended size is 100000. For more information about SQLBUFSZ, see the <i>Rocket Model 204 Parameter and Command Reference</i> .
SQLIQBSZ	SQL internal query buffer size. The recommended size is 32752.
TERMBUF	Number of terminal SNA Communications Server input/output buffers allocated. The default is 1.
TERMID	Logical terminal name associating a user ID with a particular terminal or thread for an entire run. TERMID is applicable only to threads connected through a SNA Communications Server Interface. The default is blanks.
TERMOPT	Terminal option setting. Valid settings depend on the type of terminal and access method used. The default is 0. Traffic across a SNA Communications Server (3270) network is reduced by a setting of 2 (no definite response).
TIMEOUT	At initialization, the number of seconds a thread can remain inactive before the terminal user is logged out and files closed. The maximum value is 32767. The default is 0. If the SYSOPT parameter is set for disconnection on logout, Model 204 disconnects the terminal if the teleprocessing interface supports automatic disconnect. Set TIMEOUT to the default of 0 for CRAM threads associated with IODEV=11, because these CRAM threads are restarted but not freed after a time-out.

4

Controlling System Operations (CCAIN)

In this chapter

- Overview
- System control commands
- ONLINE termination
- ONLINE monitoring
- Pseudo subtasks
- Job step return codes
- Priority scheduling
- Dynamic dispatching
- Capturing abends

Overview

Commands that control system operations are entered into the CCAIN input stream after the user environment is defined. System control commands are executed immediately. No compilation is required.

This chapter presents a summary of system control commands that can be issued by the system manager, User 0, operator, and system

administrator classes, followed by discussions on using some of these commands to:

- Terminate and monitor an Online system
- Monitor internal Model 204 special functions (pseudo subtasks)
- Control processing in response to job return codes
- Control the priority class of individual users

For a full description of each command, see the *Rocket Model 204 Parameter and Command Reference*.

System control commands

Table 4-1 describes system control commands.

Table 4-1. System control commands

Command	Purpose/comments
*DEVICE	Enables BATCH2 applications to change the values used for OUTMRL and OUTLPP parameters. (See the <i>Rocket Model 204 Host Language Interface Reference Manual</i> .)
*SLEEP	Suspends input processing for one user for a specific amount of time in Online, batch, and CMS single-user environments. The maximum time is 86400 seconds. *SLEEP can be issued by a user with User 0, system manager, or system administrator privileges. When using IFAM4, the *SLEEP specification must be longer than the application program needs to finish. If *SLEEP times out before completion of the program, IFAM4 ends and file damage can occur.
*SNAP	Generates a formatted dump for debugging purposes that shows system control blocks, user control blocks, and disk buffers.
ALLOCATE	Dynamically allocates data sets under z/OS or CMS.
AUTHCTL	Lists, views, or deletes control information for ACF2 security interfaces.
BROADCAST	Adds, changes, or removes the system LOGIN message. If the operator specifies URGENT, the message is displayed almost immediately. Otherwise, it is displayed the next time the user is at command level.
BUMP	Logs out a particular Model 204 user, all current users, all users logged in with a particular user ID, or all users who have opened a particular file. The BUMP command returns the message: M204.1124: nnnnn USERS(s) BUMPED
CHKABORT	Aborts a pending request for a checkpoint.

Table 4-1. System control commands (Continued)

Command	Purpose/comments
COPY	Copies the contents of one stream or data set to another.
CREATE	Creates a file, a temporary file group, or a permanent file group. System manager privileges are required to create permanent file groups.
CREATEG	Creates the CCAGRP file in a batch run before the first permanent group is created. The User 0 stream must contain a login command with system manager privileges. CREATEG cannot be issued from a procedure.
DEFINE	Specifies or defines the characteristics of a data set, printer, process, sequential I/O stream, permanent group, field, or punched output. System manager privileges are required for DEFINE STREAM. Any user can use DEFINE FIELD. All other forms of DEFINE require system administrator privileges. DEFINE DATASET can precede User 0's parameter line in the CCAIN input stream. It is required when using the RESTART facility.
DELETE	Deletes a procedure, temporary file group, permanent file group, or field. System manager privileges are required to delete a permanent file group. Any user can delete a temporary file group or field.
DISPLAY EW	Displays early warnings that have been applied to the Model 204 ONLINE module. DISPLAY EW requires system administrator privileges.
DUMPG	Copies the CCAGRP data set to a sequential data set on magnetic tape or on a direct access device. DUMPG is used in periodic backups and in conjunction with RESTOREG when recovering from a system failure if the Checkpoint and Roll Forward facilities are not used. Never issue an OPEN command for CCAGRP before issuing DUMPG. System manager privileges are required to issue DUMPG.
ENQCTL	Examines or modifies the status of a list of jobs contained in each file (enqueueing list) that have control of the file. Indiscriminate use of ENQCTL can affect the integrity of a shared DASD by removing list entries for active systems or jobs. System administrator or operator privileges are required to issue ENQCTL.
EOD	Specifies end-of-day and requests that all users log out. Only a user with system manager privileges can log in after all users are logged out. EOD cannot be issued from a procedure. (See the section "ONLINE termination" on page 117.)

Table 4-1. System control commands (Continued)

Command	Purpose/comments
EOJ	Specifies end-of-job and initiates Model 204 termination. Active users are forced off. File integrity might be damaged. EOJ normally appears at the end of User 0's input stream. Omission causes a user restart and sets the job step return code to 6. If the login feature is used, an Online user with system manager privileges can issue the EOJ command from a terminal.
FREE	Releases a data set and its set of attributes (template) that were previously allocated by the ALLOCATE command or in the JCL used to start up Model 204. A data set cannot be freed if it is currently in use or if it is a Model 204 system data set. System administrator privileges are required to issue the FREE command unless the ALOCPRIV parameter allows the privilege to other users.
HALT	Suspends the reading of User 0's input and provides a means of communication between the console operator and ONLINE. HALT affects only User 0. Online users continue to be served. Note that Model 204 automatically terminates when all of User 0's input is read. HALT writes a message to the operator's console and causes User 0 to wait for a reply from the operator. Replies can be those specified in the HALT command, an operator command (see the section "ONLINE termination" on page 117), or a reiteration of the message. WAITING is the default message, if no message is specified in the HALT command.
IFAMCLOSE	Closes the Host language IFAM2 channel. Before IFAMCLOSE is accepted, IFAM2 must be completely drained by issuing the IFAMDRAIN command. IFAMCLOSE is useful to free any IFAM2 programs that are waiting as the result of an IFAMDRAIN command and to allow the channel name to be switched. System manager privileges are required for issuing IFAMCLOSE.
IFAMDRAIN	Halts a Host Language IFAM2 thread. All idle threads are immediately halted. Any threads in use are started, if they were halted, and allowed to process until a IFDTHRD or IFFNSH call is issued. System manager privileges are required for issuing IFAMDRAIN.
IFAMFORCE	Forcibly terminates a Host Language IFAM2 thread. IFAMFORCE is useful in obtaining a clean checkpoint and to resolve thread allocation deadlocks (two programs each reserve one thread and are then unable to obtain a second). Forcing threads that are updating files can render a file logically inconsistent. System manager privileges are required for issuing IFAMFORCE.

Table 4-1. System control commands (Continued)

Command	Purpose/comments
IFAMHALT	Temporarily halts a Host Language IFAM2 thread. IFAMHALT is useful to improve User Language performance and to preserve an IFAM2 state for examination. Programs can time-out (ABEND 522) if they are halted too long. IFAMSTART restarts threads halted by IFAMHALT. System manager privileges are required for issuing IFAMHALT.
IFAMOPEN	Opens a specified Host Language IFAM2 channel. IFAMOPEN used in conjunction with IFAMCLOSE is useful to change the IFAM2 CRAM channel name. System manager privileges are required.
IFAMSTART	Restarts the Host Language IFAM2 facility or a specified IFAM2 thread. System manager privileges are required for issuing IFAMSTART.
IFAMSTAT	Displays the status of a Host Language IFAM2 thread. System manager privileges are required for issuing IFAMSTAT.
LOGCTL	Modifies file or group entries in the password table. Password, privileges, user class, field security levels, and terminal list can be modified. System manager privileges are required for issuing LOGCTL.
LOGFILE	Lists file entries in the password table, including privileges but not passwords. System manager privileges are required to issue LOGFILE.
LOGGRP	Lists group entries in the password table, including privileges but not passwords. System manager privileges are required for issuing LOGGRP.
LOGKEY	Specifies a password table key. System manager privileges are required for issuing LOGKEY.
LOGLST	Lists user ID entries in the password table, including privileges but not passwords. System manager privileges are required for issuing LOGLST.
MONITOR	Checks the operations of an Online system and provides statistics for SNA Communications Server (formerly VTAM) interfaces and application subsystems. (See “Statistics and monitoring of pseudo subtasks” on page 133.) System administrator privileges are required.
MSG	Sends messages between users and between users and the operator.
MSGCTL	Specifies the actions Model 204 takes when issuing an error or informational message. MSGCTL is useful to write error messages or classes of error messages to the audit trail or journal data sets. (See “Job step return codes” on page 134.) System administrator privileges are required to issue MSGCTL.

Table 4-1. System control commands (Continued)

Command	Purpose/comments
OFFLOAD	Manually starts an offload at any time during system operations. OFFLOAD is available only to RING output streams and is useful when AUTOOFFLOAD of DEFINE STREAM is set to a high value and system shutdown may occur before offloading starts. System manager or operator privileges are required to issue OFFLOAD.
PRIORITY	Assigns a user to a priority class or displays information about priority assignments. (See “Priority scheduling” on page 135.) System administrator or operator privileges are required to issue PRIORITY.
REACTIVATE	Reactivates all or specified terminals that have encountered severe I/O errors and have been taken off the polling list. REACTIVATE restarts the user and requires a login to continue. REACTIVATE applies to IODEV=15, IODEV=25, IODEV=33, and IODEV=35. System administrator or operator privileges are required to issue REACTIVATE.
REGENERATE	Restores Model 204 files when storage integrity is lost because of a hard system error, such as a disk head crash. REGENERATE can be used only within a single-user (batch) run. System manager privileges are required to issue REGENERATE.
RESTART	Restarts Model 204 and, optionally, recovers files after a system failure. RESTART is used in conjunction with the Checkpoint, Roll Back, and Roll Forward facilities. If system recovery facilities are used, the Online job must contain a RESTART command in CCAIN. System manager privileges are required to issue RESTART.
RESTOREG	Restores the CCAGRP file from a sequential data set. RESTOREG is used in conjunction with DUMPG when recovering from a system failure where Checkpoint and Roll Forward facilities are not used. System manager privileges are required for issuing RESTOREG.
START	Makes a file or group available to users and activates an application subsystem. System administrator privileges are required to issue START.
STATUS	Lists information on file recovery and current user IDs. When issued by the operator, STATUS has the effect of LOGWHO by listing all current users, user IDs, terminal IDs (including access method), open files, dead terminal lines, and threads that can be reactivated. System manager or operator privileges are required.
STOP	Makes a file or permanent group unavailable to users or stops an application subsystem. System administrator privileges are required to issue STOP.

Table 4-1. System control commands (Continued)

Command	Purpose/comments
TMASKUPDATE	Updates all the password table terminal lists at once. System manager privileges are required.
VIEW	Displays current settings of Model 204 parameters or sets of parameters. System manager privileges are required to issue VIEW ERRORS. All other options can be viewed by any user.
VTAMOFF	Closes the SNA Communications Server Interface. System manager or operator privileges are required to issue VTAMOFF.
VTAMON	Opens the SNA Communications Server Interface. System manager or operator privileges are required to issue VTAMON.
WARN	Sends a message immediately to any active user. System manager or operator privileges are required.

ONLINE termination

Caution: To avoid damaging the integrity of a Model 204 file, do not cancel the Model 204 program with the CANCEL command from the operator's console once the Model 204 program has begun to run.

Manual termination

The following steps, initiated by a user with system manager privileges, terminate ONLINE safely, if the login feature is used.

To manually terminate ONLINE:

1. Issue the EOD command without arguments or with the ON argument to notify users that ONLINE is terminating.

The following message is displayed to users when they are at command level:

```
*** M204.1028: PLEASE LOGOUT AND HANG UP
```

If User 0's input stream does not contain a login command, the following message is displayed on the operator's console when the last active user logs out:

```
*** M204.0354: ALL USERS ARE LOGGED OUT
```

After EOD is issued, only a user with system manager privileges is allowed to log in to Model 204. EOD or EOD ON can be canceled by issuing EOD OFF.

2. Issue the LOGWHO command to verify that all users have closed their files and logged off.

A list of active user IDs, their terminal IDs (indicating whether SNA Communications Server or IUCV access methods are being used), and currently open files is displayed.

3. Issue the EOJ command to terminate the job.

EOJ must be confirmed before it is executed by a Y response to the query issued by Model 204:

```
M204.1076: DO YOU REALLY WANT TO END THE RUN?
```

The default response is Y if the system manager's PROMPT parameter includes the 16 option.

Time-out termination

The following commands in User 0's input causes automatic system termination at the specific time:

```
*SLEEP number-of-seconds-ONLINE-is-available
      (User 0 suspended)
EOD
*SLEEP number-of-seconds-for-user-logout
EOJ
```

Operator control

Using the following commands in User 0's input require operator interaction:

```
HALT n,message1,m,reply1
EOD
HALT n,message2,m,reply2
EOJ
```

Where

- *n* is the number of characters in the message
- *message1* is the text of the first message
- *m* is the number of character in the reply
- *reply1* is the text of the first response

The system responds:

```
xx jobname ** 000 ** message 1
```

Where

- *xx* is a message number assigned by the system
- *jobname* is the name of the activity

Take the following steps:

1. The operator must make note of the request number (xx) when ONLINE first comes up and *message1* appears on the console.
2. The operator replies to *message1* with *reply1* when it is time to terminate ONLINE.

This causes processing of the EOD command and notifies the Online users.

3. The operator replies with *reply2* when *message2* is received and is displayed:

ALL USERS ARE LOGGED OUT.

This causes the EOJ command to terminate ONLINE without confirmation by requesting a checkpoint (if checkpoint logging is enabled) before termination processing starts. After checkpoint occurs or times out, termination proceeds.

The following sections explain replies to the HALT messages and EOD processing.

HALT processing

Replies to the HALT messages issued by User 0 can be:

- Responses coded in the HALT command
- Message itself, if no reply is specified
- WAITING (default message), if no message is specified
- Special operator commands, which result in execution of the command and redisplay of the message:

BROADCAST
BUMP
MSG
OFFLOAD
PRIORITY
REACTIVATE
STATUS
VTAMOFF
VTAMON
WARN
*SNAP

EOD processing

The following considerations apply to EOD processing:

- EOD processing is impacted by the OFFLOAD process.

The CLOSE=AUTO option indicates that the data accumulated in ring members since the last completed offload process must be offloaded during close processing. Unless EOD processing has begun and the CLOSE=AUTO parameter has been specified, the offload stream is closed whenever there are no further ring members to offload. The offload stream in this instance is left open so that the contents of all remaining full members can be offloaded.

- If the offload stream is open when EOD processing begins, it remains open even if the offload process enters an idle state before the stream is closed. This allows the final offload to continue writing to the same output data set.

ONLINE monitoring

You can monitor the status and performance of an ONLINE system, including SNA Communications Server interfaces, with the MONITOR command. Syntax for the MONITOR command is in the *Model 204 Parameter and Command Reference*.

MONITOR provides a series of formatted or unformatted displays of system and user statistics about all or a selected set of active users. You can update the displays on a periodic basis, but do not update them more frequently than every five seconds.

Unformatted displays

Unformatted displays show all the nonzero cumulative system or user statistics in a format similar to that provided in the journal:

- Series of terms appears in name=value format.
- Journal header is not displayed.
- Each user is identified by user number.
- Account name appears at the beginning of the user's display.

Formatted displays

Formatted displays show basic system information, followed by specified user information in labelled columns.

Basic system information includes the number of:

- Active users (USR)
- Active servers (SVR)
- Open buffers (BUF)
- Open files (FLS)
 - A file open as part of one or more permanent groups is followed by an

asterisk (*).

- Two pound signs (##) indicate that the user has more files open than can fit into the internal buffer used to hold open file names.
- System percentage of CPU time (PCPU).
- Statistics from the last system performance line written to the journal. If performance monitoring is active and performance lines are being written to the journal, all statistics are shown. Otherwise, only open buffers, open files, and system PCPU are shown.

Specified user information is described in the next section.

User information in formatted displays

User information is viewable in the following types of formatted displays:

- Basic display (default)
- User since-last statistics
- User performance data
- Open files

The following sections describe these types of formatted displays.

Basic display

USER	SVR	USERID	P	CUR	SLICE	AGE	FUNC	CNCT	CPU	SEQIO	QUE	WT	FLGS
0		SAM	S	48	0.075				0 0.041	3	BLKO	04	40
1	1	MARY	S	49	0.075		EVAL	266	0.152	36	REDY		

where:

- *USER* is the user number.
- *SVR* is the server number.
- *USERID* is the user ID.
- *P* is user priority (P).
- *CUR* is current priority.
- *SLICE* is current CPU time slice.
- *AGE* is user pre-aged priority.
- *FUNC* is the last function to set the since-last parameter.
- *CNCT* is connect time.
- *CPU* is CPU time.

- *SEQIO* is the sequential I/O operations performed.
- *QUE* is the current scheduler queue.

Table 4-2. Possible values for QUE

QUE value	Means user is...
RUNG	Running
BLKI	Blocked in server
BLKO	Blocked out of server
WTSV	Waiting for server
REDY	Ready or running
SWPG	Being swapped in or out
OFFQ	Invisible to the scheduler, off all scheduler queues. This typically occurs when the user is at command level
WTUS	Inactive

- *mnWT* is the internal Model 204 wait type.

Table 4-3 lists valid wait types.

Table 4-3. Valid wait types

Flag	Description
00	Unspecified
01	Disk I/O
02	Sequential user I/O — output
03	Sequential user I/O — input
04	Operator's console input (WTOR)
05	Restore reads (BSAM)
06	Dump writes (BSAM)
07	Enqueue waits (record or noncritical resource)
08	Buffer waits
09	Wait almost forever (deactivated user)
10	Waits for a pseudo subtask, or waits for reactivation (hung terminal)
11	Waits for an IFAM2 or IFAM4 call
12	Waits for a wakeup
13	Server I/O
15	Writes to the journal data set, CCAJRNL

Table 4-3. Valid wait types (Continued)

Flag	Description
16	Checkpoint writes to CHKPOINT data set
17	Waits for check on previous write
18	Waits for a checkpoint DECB to free up or waiting for multiuser output arbitration
19	Waits for a checkpoint request
20	Waits for a checkpoint completion
21	Wait forever (dead user)
22	VSAM or sequential file input
23	User waiting due to excessive login failures
24	Waits for exclusive control of critical file resource
25	Waits for share control of critical file resource
26	Waits for SNA Communications Server buffer, input or output
27	Waits for interprocess input
28	Waits for interprocess output
29	User waiting for the security interface
30	User(s) waiting, with \$WAIT(nn,'SWAP'), for a user-defined ECB to be posted by a \$POST function from a different user.
31	User(s) waiting, with \$WAIT(nn,'NOSWAP'), for a user-defined EDB to be posted by a \$POST function from a different user.
32	User waiting for DB2 subtask
33–36	Reserved for future development
37	User waiting for recovery to complete
38	Reserved for future development
39	Reserved for future development
40	Waiting for an MQSeries subtask to become available
41	Waiting for a MQ subtask to run
42	MQSeries MQGET operations with wait time specified.
43	ECF to load or delete a module.
44	External module to become free.
45	ECF subtask to become free.
46	External module to run.

Table 4-3. Valid wait types (Continued)

Flag	Description
47	User(s) waiting, with \$WAIT('CPQZ'), for the extended quiesce ECB to be posted by the successful completion of a Model 204, system-wide checkpoint.
48	User(s) waiting, with \$WAIT('QZSIG'), for the end of extended quiesce.
49	User(s) waiting, at end of extended quiesce, for count of \$WAIT('CPQZ') and \$WAIT('QZSIG') users to go to zero.
50	User is waiting for HSM to recall an archived data set.

- *FLGS* is the combined value of the status flags.

Status flags are the hexadecimal sum of the values listed in Table 4-4.

Table 4-4. Status flag values

Flag	Description of wait...
40	User's wait is swappable. If another user needs this user's server area, the waiter may be written out.
20	User is waiting for an internal ECB (that is, the ECB is posted only by Model 204, not by the operating system).
8	More than one user is allowed to wait for the ECB that this user is waiting for.
4	User can be bumped.
2	User can be interrupted with an urgent message.
1	A time limit was specified for the user's wait.

User since-last statistics

User since-last statistics (SL option) show current activity.

If the user is between activities or performing an activity that does not initialize since-last statistics, the display is cumulative.

Statistics displayed are CPU, DKRD, DKWR, UDD, OUT, SLIC, FINDS, RECDS, PCPU, RQTM, SUBSYSTEM, PROC-FILE, PROC, and the pseudo subtask's user ID, account name, and user number. (Refer to Table A-1 on page 548 for statistic definitions, and to the section "Pseudo subtasks" on page 132.)

User performance data

The PERFORMANCE option shows data last written to the journal. The following information is included:

- Number of samples used to compute percentages
- percentage of time the user is in each of the scheduler queues

See “Priority scheduling” on page 135.

Open files

This display is a list of files a user has open.

MONITOR examples

These examples show several variations of the MONITOR command.

The following command specifies a single nonrepeating formatted display of basic information for the system and all active users:

```
MONITOR
```

The following MONITOR ACTIVE command specifies a single nonrepeating formatted display of basic information for the system and active users, excepting OFFQ users. OFFQ is a state-of-queue status meaning that the Scheduler does not have to evaluate those users not in the queue. This command is used to reduce the amount of terminal output.

```
MONITOR ACTIVE
```

The following commands specify a display of the cumulative system statistics, which is updated every minute:

```
MONITOR STATISTICS          60
MONITOR STAT                EVERY 60
MONITOR SYSTEM STATISTICS   60
```

The following command specifies a single display of the cumulative user statistics for users 1, 3, and 5, if they are active:

```
MONITOR (1,3,5) STATISTICS
```

The following commands specify a single formatted display of basic system information and basic, since-last, performance, and file information for each active user. Pseudo subtask’s user ID, account name, and user number are displayed for SL (since-last) statistics:

```
MONITOR          ALL
MONITOR          USERS ALL
MONITOR          BASIC SL PERFORMANCE FILE
MONITOR          SL FILE PERF
```

The following commands specify a display of basic formatted information for the system and users 1, 2, 5, and 7, if they are active. The display is repeated

continually 75 seconds after it completes. The user must press Enter to display the refreshed version, or enter *CANCEL to terminate the display:

```
MONITOR (1,2,5,7)          75
MONITOR (1,2,5,7)          BASIC EVERY 75
```

The following command specifies an unformatted display of the following information:

- Number of active users
- Number of compactions of the record locking table
- Current number of bytes used in the record locking table
- Current high-water mark for the number of bytes used in the record locking table
- Current number of headers used in the record locking table
- Current high-water mark for the number of headers used in the record locking table
- Required LRETBL setting (high-water mark) for the current system load

```
MONITOR ENQ
```

The following command specifies a formatted display of the number of pages from tables of each file that are currently located in the disk buffers:

```
MONITOR DISKBUF
```

The following command specifies information about SNA Communications Server's use of output buffers and waits for threads:

```
MONITOR VTAM
```

MONITOR LINK example

The following command specifies information about a network entity:

```
MONITOR LINK HEADQTRS
```

For more information about the MONITOR command for network monitoring, see *Model 204 Horizon: Intersystem Processing*.

In this example, HEADQTRS is the specified network entity for which the Online system operation is being monitored. The following statistics are provided:

LOCAL ID	MAXSES	BNDSES	IBWTS	OBWTS	CONVS	FLGS	TRAN/PROTO
-----	-----	-----	-----	-----	-----	-----	-----
BOSTACB	4	2	0	0	2	A	VTAM/LU62

USER	PROCESS	SENDS	RCVS	FLGS	PROCESSGRP
---	---	---	---	---	---
3	PROCESS1	5	4	BI	DENVER
4	PROCESS1	5	4	BI	DENVER
					OBSOLETE

Where

- *LOCAL ID* is the SNA Communications Server ACB name (BOSTACB).
- *MAXSES* is the maximum number of sessions allowed on this link (4).
The limit is established by the SESSIONS parameter of the DEFINE LINK command.
- *BNDSES* indicates the number of currently bound sessions (2).
- *IBWTS* indicates the number of input buffer waits (0) by inbound conversations.
The input buffer availability is controlled by the actual number of input buffers established by the INBUFNO parameter of the DEFINE LINK command. An input buffer is used on the inbound side when a conversation is being established. If IBWTS is high, and you observe delays while opening inbound processes, then increase INBUFNO.
- *OBWTS* indicates the number of output buffer waits (0).
Note that OBWTS is always zero when the communication protocol is LU 6.2 (output buffers are not used).
- *CONVS* indicates the number of active conversations (2).
- *FLGS* indicate the status of the link status (A). Possible values for FLGS are:
A — Active
S — Stopped
C — Closed
- *TRAN* indicates the transport mechanism: SNA Communications Server or terminal (TERM). In this case TRAN is SNA Communications Server.
- *PROTO* indicates the communication protocol (LU 6.2). Possible values for PROTO are:
MSTR — Master (for TPROCESS)
XFER — Transfer (for TPROCESS)
IMS61 — IMS LU 6.1
LU 6.2 — LU 6.2

The second line of statistics is a detail line for each bound session.

- *USER* provides the external user number (3, 4). A blank occurs if a conversation is not active.

- *PROCESS* indicates the process name (PROCESS1). If a conversation is not active, a blank appears.
- *SENDS* indicates the number of physical sends for each conversation (5, 5).
- *RECVS* indicates the number of physical receives for each conversation (4, 4).
- *FLGS* indicates the status of the session (BI, BI). The session status flags are:

B	bound
F	first speaker
I	inbound
O	outbound
L	local bid in progress
R	remote bid accepted

- *PROCESSGRP* provides the process group name (DENVER). In this example, user 3 is running with an OBSOLETE process group: the process group was stopped and a DEFINE PROCESSGROUP command was issued to change the process group definition for subsequent usage. User 4 is running with the most recent version of the process group definition.

MONITOR PROCESSGROUP example

The following command specifies reports concerning a specific process group:

MONITOR PROCESSGROUP

In this example, two PROCESSGROUP reports are produced. The first report is for an active conversation. The second report reflects the stopping and redefinition of the active conversation, which is allocated to another conversation (OBSOLETE with the same process name). Until the conversation with the obsolete process group ends, separate reports are produced for each process group to reflect the more current set of attributes.

MONITOR PROCESSGROUP DENVER

```

LINKNAME REMOTEID BNDSES INCONVS OTCONVS FLGS
-----
HEADQTRS DENVACB      2      1      0 A

USER  PROCESS  SENDS RECVS FLGS
-----
   4  PROCESS1    5    4  BI

```

MONITOR PROCESSGROUP DENVER

LINKNAME	REMOTEID	BNDSES	INCONVS	OTCONVS	FLGS
HEADQTRS	DENVACB	2	1	0	AX

USER	PROCESS	SENDS	RECVS	FLGS
3	PROCESS1	5	4	BI

Where

- *LINKNAME* names the associated link (HEADQTRS).
- *REMOTEID* provides the Remote LU name (DENVACB).
- *BNDSES* indicates the number of bound sessions (2).
- *INCONVS* indicates the number of active inbound conversations (1).
- *OTCONVS* indicates the number of active outbound conversations (0).
- *FLGS* indicates the status of the processgroup (A). The processgroup status flags are:

A	Active
S	Stopped
X	Obsolete

The second line of statistics is a detail line for each active conversation.

- *USER* indicates the external user number (4).
- *PROCESS* provides the name of the program as it is known to network users (PROCESS1).
- *SENDS* indicates the number of physical sends (5).
- *RECVS* indicates the number of physical receives (4).
- *FLGS* indicates the status of the session status (BI). Status flags are:

B	Bound
F	First speaker
I	Inbound
O	Outbound

MONITOR PROCESS example

The following command specifies information about a particular process:

```
MONITOR PROCESS PROCESS1
```

In this example, two inbound conversations invoked from PROCESSGROUP DENVER (users 3 and 4) are in progress with the process named PROCESS1. The conversation ID (CID) that was assigned in the User Language OPEN PROCESS statement is PROGRAM1.

```
MONITOR PROCESS PROCESS1
```

USER	CID	PROCESSGRP	STARTED	SENDS	RCVS	FLGS	STATE
3	PROGRAM1	DENVER	88349135723	5	4	BI	RECV
4	PROGRAM1	DENVER	88349135803	5	4	BI	RECV

- *USER* indicates the external user number (3, 4).
- *CID* provides the conversation ID (PROGRAM1).
- *PROCESSGRP* indicates the name of the processgroup (DENVER).
- *STARTED* provides the Julian date and time the conversation began (88349 and 13:57:23).
- *SENDS* indicates the number of physical sends (5, 5).
- *RCVS* indicates the number of physical receives (4, 4).
- *FLGS* indicates the status of the session (BI). Status flags are

B	Bound
F	First speaker
I	Inbound
O	Outbound

- *STATE* indicates the state of the conversation (RECV). Conversation states can be:

ACCEPT
CLOSE
CONFCLS
CONFIRM
CONFSND
INITIAL
RECV

SEND

MONITOR DISKBUFF example

You can use the MONITOR DISKBUFF commands to analyze the buffer pool utilizations.

- MONITOR DISKBUFF output shows buffer usage combining above and below the bar buffers.
- MONITOR DISKBUFFG output shows buffer pool usage for only above the bar buffers.
- MONITOR DISKBUFFL output shows buffer pool usage for only below the bar buffer usage.

Use these commands throughout the day across varied types of daily and event processing to evaluate your buffer allocations.

Using MONITOR DISKBUFF commands

You can see the types of pages that are in your buffer pools at any point in time using the MONITOR DISKBUFF command. The output of this command displays the types of pages that are in buffers and how many of each type.

MONITOR DISKBUFF

FILENAME	FCT	TBLA	TBLB	TBLC	TBLD	TBLE	TBLX	*TOTAL*
-----	---	-----	-----	-----	-----	-----	-----	-----
CCATEMP	0	40	0	0	0	0	0	40
PROC1	1	0	0	0	11	0	0	12
FILETBLX	1	1	10	0	2	0	2	16
TESTZ	1	1	11	0	3	185	0	201
TOTAL	3	42	21	0	16	185	2	269

Console operator communications with Model 204 (z/VSE)

The Model 204 HALT command allows the operator to communicate with the Model 204 ONLINE configuration. The HALT message is displayed on the operator's console when the system is initialized. A response from the operator is not required.

To communicate with Model 204, follow these steps:

1. Issue the z/VSE MSG Attention Routine command to signal an ONLINE configuration:

MSG pp

where *pp* is the SYSLOG ID (such as, BG or F3) of the partition in which ONLINE is executing

2. The response to the MSG command is a message from ONLINE containing:
 - SYSLOG ID
 - z/VSE reply ID number
 - z/VSE job name of the ONLINE job stream
 - User number (always 0)
 - HALT message from the HALT command
3. The operating system issues a prompting message containing the SYSLOG ID and the z/VSE reply ID number.
4. You can issue any of the special operator commands by keying the reply ID number, at least one blank, and the command.

See the *Rocket Model 204 Parameter and Command Reference* for a detailed description of the HALT command.

In the following sample dialog, the operator requests system status information from the ONLINE configuration running in F2.

The operator keys:

```
MSG F2
```

The Online responds:

```
F2 - 002 ONLINE *** 0 *** MODEL 204 IS AVAILABLE
F2 - 002
```

The operator keys:

```
2 status
```

The Online responds:

```
F2 - 002 system status information ...
```

Pseudo subtasks

A pseudo subtask (PST) is an internal Model 204 process that performs special functions outside of normal user processing, such as checkpointing and I/O operation control. Asynchronous external events such as expiration of the checkpoint timer tend to trigger a pseudo subtask.

There are no restrictions on which internal Model 204 functions and structures a pseudo subtask can use. A user with system manager privileges specifies the pseudo subtasks processed and the resources used during a Model 204 run.

Pseudo subtasks have their own internal Model 204 structures, pushdown lists, and copies of certain server areas to enable them to take complete advantage of Model 204 facilities. Each pseudo subtask requires approximately 1500 bytes of additional storage.

Statistics and monitoring of pseudo subtasks

Pseudo subtask statistics are gathered on CPU time and elapsed connect time. The information is viewable through the MONITOR (USERS option) command, partial, and final statistics displays.

Each pseudo subtask has a user ID, account, and user number for external viewing. The user IDs are listed in Table 4-5.

Table 4-5. Internal pseudo subtasks

Task name	Handles...
CHKPAWW	Wait for preimage writes (31-bit only)
CHKPPST	Checkpointing
CHKPTIMO	Time limit for quiescing users
CHKPTIMR	Checkpoint every CPTIME minutes
CMSVTAM	Asynchronous SNA Communications Server exits for CMS/SNA Communications Server; handles loss of IUCV connection to VT204
OFFLOAD	Offload of ring streams
PRT-PERF	Logging of performance data
PRT-PART	Printing of partial statistics
VTNTDIED	Notification of SNA Communications Server crashing
VTNTERRS	SNA Communications Server LOGON errors for NTO devices
VTNTREAD	Input from SNA Communications Server NTO devices
VT62ACBS	Special services for Horizon links
VT62CLPS	CLOSE LINK FORCE processing
VT62RAPS	Input from Horizon partner processes
VT75DIED	Notification of SNA Communications Server crashing
VT75ERRS	SNA Communications Server LOGON errors for 3270 devices
VT75READ	Input from SNA Communications Server 3270 devices

Messages

The pseudo subtask feature issues messages when:

- Pseudo subtask starts processing
- Pseudo subtask completes processing
- Specified number of terminals (NOTERM) in an interface group is too large. The number displayed indicates the actual number of threads available.

When this occurs, the interface is initialized and awakened, not left in a hanging state.

Messages are issued for SNA Communications Server NTO and SNA Communications Server 3270.

Job step return codes

Job step return codes provide the operating system with information that determines whether a step is executed or aborted. The handling of individual job step return codes is installation specific. Many installations choose to ignore all return codes in the range 0–95 for ONLINE runs.

The job step return code is available from z/OS and z/VM operating systems and in the following audit trail AD line message under all operating systems:

```
*** M204.1075: TERMINATION COMPLETE. RETURN CODE=nnn
```

Using MSGCTL command with SYSOPT parameter to produce an abend

Using the MSGCTL command you can set the precise return code when you want the error to return. You can set the error to any non-zero return code. If the SYSOPT parameter is set to include 40, any non-zero return code from a Model 204 message will generate an abend without a dump at termination.

Table 4-6 summarizes the most common return codes issued for Model 204 steps.

Table 4-6. Job step return codes

Return code	Meaning
0	Successful completion of the job step
6	User restarted
8	Warning
16	Error during User Language request compilation
20	Error in file operation
24	Error during FLOD compilation
32	Maximum number of errors exceeded
40	Error during User Language request evaluation
44	In a batch run, an uninitialized or physically inconsistent file is opened. Expect a return code of 44 in some circumstances, such as when a CREATE and INITIALIZE is done. OPEN after CREATE (and before INITIALIZE) produces a return code of 44 indicating that the file is not yet initialized.
48	Table A, B, C, D, E or X is full

Table 4-6. Job step return codes (Continued)

Return code	Meaning
52	Error in recovery or restart operation
56	Serious error during User Language request evaluation
60	Scratch file full or run canceled with dump
64	Error during FLOD evaluation
72	Error due to Table D inconsistency (message 0447)
80	Error during system initialization
88	Error during system termination
96	Serious error in Model 204; a SNAP is written to the CCASNAP data set
100	SNAP production failed
104	I/O error on CHKPOINT or CCAJRNL file

Conditional job control

Under z/OS and z/VSE, you can code conditional job control statements based on job step return codes by coding the operating system COND parameter. Please consult IBM documentation regarding the COND parameter for details.

Priority scheduling

The system manager uses the PRIORITY command and the PRIORITY parameter on a user IODEV line in CCAIN to allocate Model 204 resources to users based upon their relative service requirements.

Users can be in one of three priority classes: LOW, STANDARD, or HIGH. In general, HIGH priority users receive service sooner than STANDARD priority users, and STANDARD priority users receive service sooner than LOW priority users.

You can also specify ranges for the PRIORITY command.

PRIORITY command and PRIORITY parameter syntax

LOW, STANDARD, and HIGH use the following default ranges:

Priority	Cur	Min-Max
LOW	016	001-047
STANDARD	048	032-079
HIGH	096	080-127

**PRIORITY
command
syntax**

Use this syntax when issuing the PRIORITY command:

```
PRIORITY [userno [,LOW|STANDARD|HIGH]]
```

or

```
PRIORITY [userno [,cur|,(cur,min,max)] [,keyword=value]]
```

or

```
PRIORITY userno, CANCEL
```

**PRIORITY
parameter
syntax for
IODEV threads**

Use this syntax when specifying a priority on an IODEV thread defined in CCAIN:

```
PRIORITY=[LOW | STANDARD | HIGH]
```

or

```
PRIORITY=(cur[,min,max]) [,keyword=value]
```

where:

- *userno* specifies the user number to modify or display. If no other parameters are specified, the specified user's current settings are displayed.
- *cur* specifies a new current priority, 0-253, for the specified user.
A user's current priority is not required to be within his range. However, as the user ages, the current value will rise or fall until it falls within the given range.
- *min* specifies a new minimum priority, 1-253, for the specified user.
- *max* specifies a new maximum priority, 1-253, for the specified user.
- *keyword=value* changes the value assigned to one of the keywords listed in the table below; for example `IOSLICE=60`. To reset values to system defaults, specify a null value. See "Examples" on page 138.

The following table lists the recognized command and parameter keywords. A command keyword is used on the PRIORITY command, and a parameter keyword is used when setting priority on an IODEV line in CCAIN.

Command keyword	Parameter keyword	Specifies
IOSLICE	UIOSLIC	CPU milliseconds allowed while user is I/O bound.
CPUSLICE	UCPUSLIC	CPU milliseconds allowed while user is CPU bound.

SLCWAIT	USLCWAIT	Sleep time in milliseconds, invoked each time a user reaches minimum priority level.
SLCMAX	USLCMAX	Number of Stop-Loop-Checks (SLCs) before CSLICE invoked (max=65535). Reducing this number increases the accuracy of the slice interval, However, CPU overhead increases. Note: SLC is an internal Model 204 routine that prevents infinite user loops.

- CANCEL will end the user's current request. The error procedure will be invoked if the user is in a subsystem.

Viewing changes in priority

When you issue the PRIORITY command to change a user's priority, the values existing prior to the command are shown. To verify the new values, issue the PRIORITY command again, as shown in the following example.

```
PRIORITY 9,33
  USER USERID P CUR,MIN-MAX  SLICE  IOSLICE CPUSLIC MAX SLCWAIT SERV
CPU
    9 USERXX  H 095,080-127   0.000I   0.030   0.010  50   0.00    1
0.000
PRIORITY 9
  USER USERID P CUR,MIN-MAX  SLICE  IOSLICE CPUSLIC MAX SLCWAIT SERV
CPU
    9 USERXX  H 033,080-127   0.000I   0.030   0.010  50   0.00    1
0.000
```

Duration of PRIORITY assignments

Once a priority has been assigned, that priority remains in effect until it is changed by another PRIORITY command or until you log out of Model 204. On logoff or restart, all priority parameters are reset to either their user default values (set on IODEV) or their system values.

Setting PRIORITY to 0

If a user priority is set to zero, that user will no longer be dispatched. Instead the user remains logged in, but is suspended during evaluation. The suspension can occur at command level or at the bottom of a FOR loop. That user must be reset to a non-zero priority to continue. (Exception: If the user is updating or holds critical resources, they will be allowed to run to COMMIT or END of request before being suspended.)

Examples

The following priority command changes User 38's current priority to 100, minimum to 80 and maximum to 120. The value of IOSLICE, for this user only, is also changed to 60.

```
PRIORITY 38, (100,80,120), IOSLICE=60
```

To reset argument values back to system defaults, specify a null value. For example to change User 38 back to the system default value for IOSLICE:

```
PRIORITY 38, , IOSLICE=,
```

The trailing comma is required and indicates the null value. The double commas after 38 indicate that current priority or priority range has been omitted.

Delaying work

When necessary, the system administrator can delay work to accommodate other, higher priority work. For example, if low importance BATCH2, HORIZON or other IODEV threads are running and work of higher importance must be run the low importance threads may be set to an extremely low and narrow priority range and SLCWAIT and SLCMAX values may be added.

Let's say you want to drastically increase the elapsed time a BATCH2 thread requires to run, essentially run it as a very low priority, background task. If the BATCH2 thread is user 59, the following PRIORITY command will allow that thread to run once every 2 seconds (SLCWAIT) for 100 milliseconds (IOSLICE). After 100 milliseconds or when the thread issues any kind of wait for an external event—disk I/O, READ SCREEN, server swap, record/resource locking conflict resolution, pause, and so on, the thread will wait for two seconds before being dispatched again:

```
PRIORITY 59, (5,5,5), IOSLICE=100, SLCMAX=1, SLCWAIT=2000
```

Users may also be suspended with the PRIORITY command. If you suspect a runaway application, but want to confirm before bumping the user, you could suspend that user by setting priority to zero:

```
PRIORITY 72, 0
```

Note: Setting a user to low or zero priority, however, must be done with care. Record locks continue to be held for a LOW or zero priority user. Other users, who need to process those records, may be blocked.

PRIORITY command output

The output of the PRIORITY command includes a header line which indicates the meanings of the statistics in the user lines that follow. User and server numbers occupy up to five characters.

If PRIORITY is entered with no parameters, all users are displayed. You may abbreviate the PRIORITY command to PRI.

Note that the column below labeled MAX is SLCMAX.

PRIORITY

```

USER  USERID  P CUR,MIN-MAX  SLICE  IOSLICE  CPUSLIC  MAX  SLCWAIT  SERV
CPU
  0 NO USERI S 253,032-079  0.000I  0.070   0.100  50   0.00   OUT
0.001
  1 BECKETT S 061,032-079  0.000I  0.070   0.100  50   0.00   OUT
0.013
  2 LESTER  H 127,080-127  0.000I  0.070   0.100  50   0.00    5
0.170
  3 MATSUZAK S 061,032-079  0.000I  0.070   0.100  50   0.00   OUT
0.012
  4 PENNY   H 104,080-127  0.000I  0.070   0.100  50   0.00    1
0.422
  5 WAKEFIEL H 114,080-127  0.000I  0.070   0.100  50   0.00    3
0.105
  6 VARITEK S 079,032-079  0.000I  0.070   0.100  50   0.00   OUT
0.063
  7 YOUKILIS S 079,032-079  0.000I  0.070   0.100  50   0.00   OUT
0.069
  8 PEDROIA S 079,032-079  0.000I  0.070   0.100  50   0.00    6
0.133
  9 LOWELL  S 072,032-079  0.000I  0.070   0.100  50   0.00   OUT
0.032
 11 LUGO    S 079,032-079  0.000I  0.070   0.100  50   0.00    2
0.112

```

The priority (P) column (third from the left in the previous priority display) can have the following values listed in the following table.

Priority column	Specifies...
L	Low
S	Standard
H	High
*	User defined
Z	Sleeping (priority=zero);
?	Deferred priority change in progress, which will take effect when the user issues COMMIT command, BACKOUT command, or request processing ends.

The SLICE column displays either I for I/O bound or C for CPU bound.

Consider this scenario of user scheduling

For a user, who in this example is USER11, USERID=LUGO, PRIORITY can be set as follows:

```
PRIORITY 11, (20,16,40), SLCWAIT=2000, SLCMAX=2,      X
IOSLICE=30, CPUSLICE=10
```

USER11 will begin work with a priority of 20. Any users with STANDARD priority or higher will receive CPU cycles ahead of USER11. Once USER11 has received 30 milliseconds of CPU (the IOSLICE value at which the user is declared CPU-BOUND), USER11 will be declared CPU-BOUND for the next 10 milliseconds.

At selected times during the processing (loop, FIND, FOR, and so on), the Model 204, internal, Stop-Look-Check (SLC) routine will be called on behalf of USER11 to evaluate his time slice. Since SLCMAX=2, the first SLC will skip the check of the time slice. On the second call, the usage will be examined. If USER11 has exceeded 10 milliseconds (CPUSLICE), then USER11 will be time sliced (rescheduled) and a user with higher priority is allowed to run.

Each time USER11 is rescheduled and is found to be CPU-BOUND, his priority will be lowered by 2. So, after approximately 40 (30+10) milliseconds of CPU, USER11 now has a priority of 18. USER11 continues to run and two SLC calls later, USER11 is again time sliced.

USER11 has now consumed approximately 50 (30+10+10) milliseconds of CPU and his priority is now 16. This is the bottom of his range, so the SLCWAIT parameter becomes active.

Since SLCWAIT=2000, USER11 is placed in a timed 2-second (2000 milliseconds) wait and will not be rescheduled for evaluation until this time has elapsed. If no other work is found for Model 204 to process, Model 204 will enter an operating system wait until USER11 or another user becomes eligible to be dispatched. In the meantime while Model 204 waits, the operating system may schedule other address spaces, virtual machines, or partitions.

CUSTOM=(9) setting

The CUSTOM=(9) setting suppresses all output from all forms of the PRIORITY command.

Scheduler performance parameters

Use the scheduler performance parameters listed in Table 4-7 to tune the scheduler's performance characteristics. You can set these parameters on User 0's parameter line or reset them if you are a user with system manager privileges.

Use the VIEW CWAIT command to examine scheduler parameters

These parameters are described in the *Model 204 Parameter and Command Reference*.

Table 4-7. Scheduler performance parameters

Parameter	Description
AGEINCR	Internal priority increment associated with aging promotion
AGEINTVL	Real-time interval in milliseconds a user must wait before being promoted by the aging feature
AGESCAN	Allowable real-time interval in milliseconds between scans of wait queues performed by the aging logic
AGESLICE	CPU time-slice increment associated with aging promotion
CFRWPCT	Specifies the percentage of servers, NSERVS, that can be occupied by non-swappable users waiting for exclusive access to a critical file resource (CFR).
CPUSLICE	CPU time-slice allotment for CPU-bound users
IOSLICE	CPU time-slice allotment for non-CPU-bound users
PRIOMAX	Maximum priority that aging promotion is allowed to produce
SIOSLICE	Server real-time-slice allotment in milliseconds for non-CPU-bound users
SLICEMAX	Maximum CPU time-slice in milliseconds that immunity aging can produce
SRVSLICE	Server real-time-slice allotment in milliseconds for CPU-bound users
UCPUSLIC	CPU time slice allotment for an individual user
UIOSLIC	I/O time slice allotment for individual user
USLCMAX	Maximum CPU time slice for an individual user
USLCWAIT,	Sleep time at minimum slice priority
WAITSCAN	Real-time interval allowed to pass between wait queue checks

Dynamic dispatching

Dynamic dispatching is the internal process used by Model 204 to alter user priorities. Individual users within the same priority class are not necessarily treated equally, because internal priorities can temporarily modify the order of processing. Model 204 adjusts the internal priority of each user, subject to the limits imposed by the user's priority class, in order to maximize the total Model 204 throughput.

Dynamic dispatching rules

The dynamic dispatching rules used by Model 204 reward users who perform terminal I/O, especially input, and discourage users who consume a lot of CPU time without performing any terminal or disk I/O.

Just prior to processing a command, Model 204 sets a user's internal priority to the central value of the user's priority class (LOW = 16, STANDARD = 48, HIGH = 96). Increases are subject to the maximum internal priority implied by the user's priority class.

- Each terminal output raises the user's internal priority by 1.
- Each terminal input raises the user's internal priority by 2.

Each time a user consumes more than a preset amount of CPU time without performing any I/O (disk or terminal), the user is judged to be CPU-bound and the internal priority is decremented by two units, subject to the minimum internal priority implied by the user's priority class.

Each time a user, previously judged to be CPU-bound, voluntarily surrenders the CPU by performing disk or terminal I/O, the user's internal priority is incremented by one, subject to the maximum internal priority implied by the user's priority class.

Balancing CPU bound and I/O bound users

The CPUSLICE and IOSLICE parameters provide additional control over CPU bound users. The default values in milliseconds are:

- CPUSLICE=10
- IOSLICE=30

Each user receives a slice of CPU time whenever the user is dispatched, that is, given the CPU. The CPU time slice is initially set to IOSLICE milliseconds. As a user consumes CPU, the amount consumed is compared to the IOSLICE value.

- A user who relinquishes the CPU and goes into a wait state, by issuing a READ SCREEN statement or by performing database I/O to Table A, B, C, or D before consuming IOSLICE milliseconds of CPU, is considered I/O bound.
- If a user has not gone into a wait state within IOSLICE milliseconds, the user is considered CPU bound.

When a user becomes CPU bound, the user's priority is decremented by two points and the CPU time slice value is changed from IOSLICE to CPUSLICE milliseconds. Until the user voluntarily goes into a wait state, the amount of CPU time provided per time slice equals the CPUSLICE value.

This improves response time for I/O bound users by preventing CPU bound users from monopolizing the CPU.

Queue aging

Queue aging, which increases a user's priority when a wait for Model 204 resources (server areas, the CPU, disk buffers, and others) occurs, sets a maximum allowed service time for lower priority requests:

- As users are aged, they are assigned a time-slicing immunity value, which also increases with time, subject to specified limits.
- An aged user's priority is not decreased until the user has consumed an amount of CPU time equal to or greater than the user's current time-slice immunity.
- Once an aged user's priority reaches a level that allows selection for running, the aged priority is maintained until the user's time-slice immunity has been exhausted. At that point, the user's priority is returned to its pre-aging value and the time-slicing immunity is removed.

Although the application of aging defeats normal priority scheduling, its impact can be minimized through careful selection of aging parameters (default settings do not allow aging to take place). Proper aging parameter settings avoid interruptions that higher priority users might experience as lower priority users receive a greater level of service before returning to a lower priority.

User 0's priority

User 0 receives the highest possible priority of any Model 204 user when a HALT command is issued. The settings of the minimum, maximum, and current priorities are set to 128, which allows User 0 the highest possible dispatching priority of any Model 204 user.

Capturing abends

Asynchronous SVC dumps can be generated and written to SYS1.DUMP data sets on Model 204 abends. Model 204 continues as soon as the pages are copied to the DUMPSERV address space. All physical I/O is done from DUMPSERV, which frees the Online system sooner.

You can enable the asynchronous dump by specifying SNAPCTL=X'05', which is a separate setting that produces a complete region asynchronous dump. For a more detailed discussion of the SNAPCTL parameter see the *Rocket Model 204 Parameter and Command Reference*.

Warning: Familiarize yourself with the memory requirements of the asynchronous dump process to ensure that enough expanded and page space is available. Memory shortage can cause severe system degradation while an asynchronous dump is processing the DUMPSERV address space. **If you use**

this setting and you have not properly configured the DUMPSERV memory requirements, you risk locking your z/OS system.

Note: If the dump is not taken, due to suppression by Dump Analysis Elimination (DAE) or a z/OS problem, the following IBM error message is written to the operator:

```
'DUMP FAILED - REASON XX'
```

The reason codes are listed in the SDUMPX macro in the *z/OS Auth Assm Service Reference LLA-SDU*. The number of the manual varies according to the release of z/OS.

5

Using the System Scratch File (CCATEMP)

In this chapter

- Overview
- Sizing CCATEMP
- CCATEMP in system recovery
- Using cache fast write
- CCATEMP statistics
- z/OS considerations
- CCATEMP in Storage feature

Overview

CCATEMP is a required temporary or permanent data set that is used as a system scratch file for a single job on one CPU.

Up to 11 CCATEMP data sets can be used in all operating system environments to accommodate a corresponding number of jobs.

In the course of a Model 204 run, CCATEMP pages are initialized and used as work areas to:

- Store temporary procedures and edit procedures
- Hold record set lists and the results of FIND statements

- Store sorted record sets
- Sort FIND and FOR EACH RECORD values
- Store temporary groups, backpage images, screen, menu and image definitions, temporary group definitions, group field tables, and application subsystem precompiled procedures
- Store back out and constraint logs for transaction back out files
- Resolve IS LIKE pattern matching syntax

This chapter provides guidelines for CCATEMP space allocation, explains the role of CCATEMP in recovery, and summarizes job control information for z/OS and z/VSE environments.

Sizing CCATEMP

The space required by CCATEMP depends on the complexity of the retrievals executed. Do not increase the size of CCATEMP unless you need to for sort work or temporary tables. Overallocating CCATEMP wastes memory and disk space.

The maximum size of CCATEMP is 16,777,215 pages.

Model 204 functions are now divided into those that require 2-byte page numbers and those that can use 3-byte page numbers. If a function can use 3-byte page numbers, then its pages are allocated above the 64K line. If all such pages are already allocated, then it is assigned a 2-byte page number if one is available.

If a function requires 2-byte page numbers, then its pages are allocated below the 64K line. It is, therefore, possible to fill CCATEMP even if there are unused pages above the 64K line.

During Model 204 termination, a message is issued that summarizes CCATEMP usage:

```
CCATEMP PAGES USED = m OUT OF n
```

Many commonly used Model 204 features and facilities affect the size of CCATEMP. When allocating space for CCATEMP, take note of the special considerations described in the following sections.

Transaction back out in recovery

When the Transaction Back Out facility (TBO) is used in recovery, extra CCATEMP space is needed for reapplication of updates from the recovery data set. Use the following formula:

```
Pages of additional CCATEMP storage = 1 page/transactions  
* maximum number of concurrent transactions.
```

Note: Large transactions may require more than 1 page.

FLOD exits

Additional CCATEMP space is required when the FLOD exit (FLODXT#) program is run. The FLOD exit feature is discussed in the *Model 204 File Manager's Guide*.

File groups

Open Model 204 file groups actively use CCATEMP and release used pages when the group is closed. If your run uses file groups, take note of the following requirements:

- At least one page is required for each user who creates a temporary file group definition or opens a permanent group.
- One scratch page is required for each permanent or temporary file group that has a field name referenced by a User Language request or IFAM call. The file group must have a BLDGFT=YES parameter setting.
- For either a batch or an ONLINE run that uses file groups, a scratch file of at least 25 pages per user is suggested. Otherwise, 20 pages are generally sufficient.
- An error message citing a file full condition for CCATEMP signifies the necessity of increasing the space allocation.

Ad hoc file groups do not build group field tables on disk.

Subsystem Management and precompiled requests

When the Subsystem Management facility (SUBSYSMGMT) is installed, CCATEMP requires additional space to accommodate the compiler tables for precompiled procedures. The data stored in CCATEMP consists of a header section and the contents of VTBL, QTBL, STBL, and NTBL. For example:

```

94 + (32 * VTBL HWM) + (12 * NTBL HWM) + (STBL HWM)
    + NFILES
    + NRMTFILE + (16 * QTBL HWM)
    + (ad hoc group FTBL space)
    + ((30 + (7 + NRMTLOCS)/8) * #groups)
    + (8 * (#fields referenced in group))
    + (the sum of the length of the names of all the
      fields referenced in the group)
    + #screens + #images
    + (unavailable-file space) + (XVAR space)

```

Where

- *HWM* refers to the high-water mark found in the audit trail's since-last compilation statistic for the indicated table.
- *ad hoc group FTBL space* depends on whether there are ad hoc scattered groups (Parallel Query Option/204) included. If no ad hoc groups are scattered, ad hoc group FTBL space is:

$$62 * (\text{\#ad hoc groups})$$

If some ad hoc groups are scattered, *ad hoc group FTBL space* is:

$$(62 + (\text{\#open files in ad hoc groups})) * (\text{\#ad hoc groups})$$

- *unavailable-file space* is the following quantity, not knowable in advance, which you need to estimate (Parallel Query Option/204 only):
- $$(4 + (\text{\#unavailable group files})) * (\text{\#groups with unavailable members})$$
- *XVAR space* is required for these User Language request elements: found sets, lists, FOR statements with a WHERE clause. The number of bytes per element depends on the file or group context, as follows:

Transaction context	Bytes required per element
Single file	8
Ad hoc or permanent group	8 * (#files in group)
Temporary group	4 + (8 * (#files in group))

The number of additional CCATEMP pages required is:

$$\text{The result of the above calculation} / (\text{PAGESZ} - 40)$$

CCATEMP in system recovery

Shared DASD

CCATEMP cannot be shared between jobs on a single CPU or placed on shared direct access devices (DASD) that might be accessed simultaneously by two CPUs. A shared DASD enqueue list protects CCATEMP in shared DASD situations. Because CCATEMP is opened as part of Model 204 initialization, you cannot use the ENQCTL command to modify the enqueue list.

After a system crash

In the event of a system crash, the following considerations apply:

- If CCATEMP is defined as a permanent file and recovery is not run, you must reinitialize or overwrite CCATEMP from outside the Model 204 environment.
- If CCATEMP is defined as a temporary file, overwrite it in a prior job step using IEBGENER (z/OS), DITTO (z/VSE), or the CMS FORMAT command to avoid the possibility of receiving an old scratch file from another job.
- If recovery is run after a system crash, Model 204 automatically performs the required maintenance on CCATEMP.

Using cache fast write

If you have cached DASD controllers (such as IBM 3990 models 3 and 6) you can use the CACHE parameter to allow CCATEMP data to be written directly to a controller's cache.

CCATEMP statistics

Statistics are kept for every file opened during the run, including CCATEMP. The number of CCATEMP pages used in a run appears in the journal (TFMX) at the end of the run.

The high-water mark of CCATEMP usage and the total number of pages allocated to CCATEMP is reported on the audit trail in the following statement:

```
CCATEMP PAGES USED=n OUT OF m
```

Handling 64-bit statistics

To support very long running Model 204 regions, Rocket Software has modified the capacity of statistical counters by increasing the size of some statistics and also exploiting 64-bit processing where appropriate. For any in-house or third-party support applications that process statistical counters you will need to review the statistics generated.

As some of the statistics fields are now double-words, check *Appendix A: Using System Statistics* for the new layout of the System, Final and Partial statistics. Also, additional Disk Buffer Monitor, MP/204, and File statistics have been updated.

Look at your in-house or third-party support applications to see if you need to make changes because of the increased length of some of the statistics and make any changes necessary to your applications, then reassemble with this new release.

If your in-house or third party support applications don't reference any of these double word statistics, then you only need to reassemble your program with the new offsets documented in this new release.

CCATEMP statistics at termination

The M204:2622 message is printed twice at termination to report the high water mark of the number of pages used in both the small model area and the expansion area.

```
M204.2622: HWM CCATEMP PAGES USED IN SMALL MODEL AREA  
           = nnnnnnn, MAX AVAILABLE = 65536
```

```
M204.2622: HWM CCATEMP PAGES USED IN EXPANSION AREA  
           = nnnnnnn, MAX AVAILABLE = nnnnnnn
```

The two CCATEMP statistics, TEMPHIE and TEMPHIS report the high water mark of pages used in the CCATEMP expansion area and in the small model page pool, respectively.

z/OS considerations

In a z/OS environment:

- The JCL must contain a DD statement for each CCATEMP data set.
- If you specify more than one CCATEMP data set for a run, you must call the first data set CCATEMP, and the unit type must be compatible with the installation's page size:

```
//CCATEMP DD UNIT=3380,DISP=(NEW,DELETE),SPACE=(TRK,80)
```

- You must name subsequent CCATEMP data sets CCATMP0, CCATMP1 through CCATMP9. The data sets are opened in numerical order during initialization. Any break in the order ends the open process. Only those data sets successfully opened are used as part of CCATEMP for the run.

Storing record set bit maps

Record set bit maps are stored in CCATEMP pages whether bit X'40' of SYSOPT2 is set or not. The X'40' setting will limit the CCATEMP page range that can hold these record set bit maps.

- When SYSOPT2 setting does not include X'40', then at any given time the bit maps corresponding to all users holding found sets of any kind must fit into CCATEMP pages in the page number range X'0000'–X'FFFF' (that is page number 0–65,535, the small model page pool) no matter how many pages have been allocated to CCATEMP.
- When the SYSOPT2 setting does include X'40', the off-restriction is removed and user found sets can be placed anywhere within CCATEMP. This includes both the small model page pool (pages 0–65,535) and the CCATEMP expansion area (page numbers 65,536 to 16,777,215), allowing for the possibility of a greater number of concurrent found sets being held by all users.

CCATEMP in Storage feature

This feature is for sites which:

- Have z/OS in 64-bit mode and real storage larger than 2-gigabytes, or
- Run z/OS in 31-bit mode and have enough expanded storage to support dataspace with no excessive paging. A **dataspace** is an address space in z/OS with none of the control structures for tasks; it is simply data.

Other operating system configurations will begin excessive paging, and serious performance degradation of the entire system is likely to occur.

If you have enough real storage on your system, you can place either or both CCATEMP file and CCASERVR file in storage. The benefit to you from activating CCATEMP in Storage depends on how much latent wait time is spent by your jobs waiting for the CCATEMP file to complete.

- There is no CYLINDER size restriction on SERVSIZ, which is limited only by available memory.
- The CCATEMP file initialization is not required with CCATEMP file in Storage feature. This significantly reduces the time required for Model 204 initialization when a large CCATEMP file is required.
- Less CPU is consumed per logical CCATEMP I/O. A CCATEMP I/O, with this feature, is logical since data is moved in-memory and not to or from an external device.
- An additional benefit is that relief is provided on I/O subsystems. Because there is less contention, channels and disks are free to perform other services and jobs run faster.
- The APSYPAGE parameter may compliment the CCATEMP in Storage feature; see the *Rocket Model 204 Parameter and Command Reference* for details of the APSYPAGE parameter.

Activating the CCATEMP in Storage feature for a job

To activate the CCATEMP in Storage feature in a job, you must set a new parameter in the USER0 CCAIN stream. This parameter is TEMPPAGE=*number-of-CCATEMP-pages-to-allocate*. You may comment out or remove the CCATEMP DD record(s), but this is not a requirement. If the TEMPPAGE parameter is set, the DDs for the CCATEMP file are ignored by Model 204, although not the operating system.

System programmers consideration

To use the CCATEMP in Storage feature, be aware that the maximum number of data- or hiperspaces and the total cumulative size of data- or hiperspaces for a given job may be controlled by the SMF IEFUSI exit.

In this exit, the parameter description shows that subwords 2 and 3 of the word 7 parameter are the keys, where:

- Word 7 subword 2 controls the maximum cumulative data- or hiperspace size
- Word 7 subword 3 controls the maximum number of data- or hiperspaces that a job may allocate

Check the IBM supplied settings and change them, if necessary.

Information regarding this exit routine is found in the *OS/390 z/OS Installation Exits Document Number SC28-1753*.

Tracking dataspaces and hiperspaces

See the MONITOR DATASPACE command in the *Rocket Model 204 Parameter and Command Reference*.

z/VSE and CCATEMP considerations

Allocating CCATEMP

In a z/VSE environment, CCATEMP must be allocated using the ALLOCATE utility before an ONLINE, BATCH204, or an IFAM1 program can be executed.

The Job Control Language statements required for the execution of the ALLOCATE program are:

```
// JOB ALLOCTMP ALLOCATE MODEL 204 SCRATCH FILE
// DLBL M204LIB,'M204.PROD.LIBRARY'
// EXTENT ,...
// LIBDEF PHASE.SEARCH=M204LIB.V210
// DLBL CCATEMP,'M204.name.CCATEMP',99/365
// EXTENT SYS001,balance of extent information
// ASSGN SYS001,X'cuu'
// EXEC ALLOCATE,SIZE=AUTO
ALLOCATE FILE(CCATEMP)
/*
/&
```

where the statement:

```
// LIBDEF PHASE.SEARCH=M204CL.PROD
```

applies to Model 204 z/VSE Version 2.1.0 or later.

Label information

When executing the ONLINE, BATCH204, or user-written IFAM1 program that uses the CCATEMP file, you must provide the following label information in the execution job stream:

```
// DLBL CCATEMP,'M204.name.CCATEMP',,DA
// EXTENT SYSnnn,balance of extent information
// ASSGN SYSnnn,X'cuu'
```

Where

- *DA* is the file organization to specify when executing ONLINE, BATCH204, or a user-written IFAM1 program.
- *SD* is the file organization to specify when running ALLOCATE.

Multiple CCATEMP data sets

If you use multiple CCATEMP data sets, the naming convention must reflect the 7-character limit. Define the files in numerical order as follows:

```
CCATEMP,CCATMP0,CCATMP1 through CCATMP9
```


6

Storing and Using File Group Definitions (CCAGRP)

In this chapter

- Overview
- File groups
- Types of file groups
- Using file groups
- Creating the CCAGRP data set
- Storing group definitions in CCAGRP
- Creating file groups
- Opening, closing, displaying, and deleting file groups
- File group passwords and privileges

Overview

File groups are used to organize cyclic data, to archive aging data, to organize independent but similar files, and to access the same file by different names.

This chapter provides general information about file groups, followed by instructions on how to create and use a data set containing file group definitions (CCAGRP).

File groups

A **file group** is a defined list of one or more physically independent but logically related Model 204 files that can be accessed dynamically as a single file. When a group is defined, an entry containing the names of the files in the group is maintained in an internal table in the permanent file group data set CCAGRP.

An individual file can be a member of several different groups and a group can contain as many as 255 Model 204 files. Most installations require only one CCAGRP file.

Commands, User Language statements, and Host Language Interface calls that refer to a group cause operations to be performed on each member file in turn.

As Model 204 processes an OPEN command for the group, it refers to the internal table, determines which files are associated with that group, and opens each of the member files.

When a FIND statement operates on a group, it acts on each of the member files in turn, resulting in a set of records that includes qualifying records from each of the member files.

Types of file groups

File groups can be permanent, temporary, or ad hoc.

Permanent groups

Permanent groups have the group name and member list stored permanently in the external file CCAGRP until explicitly deleted.

- The system manager creates and maintains permanent groups.
- Any user can open a permanent group.
- Passwords can be required to access the group.
- You cannot create or delete permanent file groups when more than one Model 204 system is running with the same CCAGRP.

Temporary groups

Temporary groups are created by an individual user with the CREATE command and can be referenced only by that user.

- Temporary group names exist only for the current Model 204 login session.
- Temporary group names are deleted when the user logs out, unless the names were explicitly deleted during the session.

Ad hoc groups

Ad hoc groups are created within a User Language request by prefacing a statement with the following:

```
IN file1,file2,...
```

All the files in an ad hoc group must be opened before the group is created.

- Ad hoc groups have no name and exist only for the current request.
- Ad hoc groups facilitate record retrieval from many files at once without requiring that a group be defined in advance.

For more information about ad hoc groups, refer to the *Model 204 User Language Manual*.

Using file groups

File groups support dynamic access to multiple, physically independent files that are separately maintained but logically related. Files and file groups can have the same name.

Accessing cyclic data

Cyclic information can be accessed through files containing data segments and then grouped together for access to all the data. An internal table relates the whole (file group) to the segments (files) without data duplication. All the member files need not exist until the group is opened.

For example, file group WEEK consists of the files SUNDAY, MONDAY, TUESDAY, WEDNESDAY (maximum eight letters), THURSDAY, FRIDAY, and SATURDAY. Each daily file is accessible directly under its own name. The union of all files is available under the name WEEK.

Changes made to current members of a group, such as adding new fields or changing field definitions, does not impact the ability to use prior versions of the files in a group. For example, if you keep the five most-recent-fiscal-year files in a group and you then add a new field to the most recent year, you have not disturbed your use of the other files in the group or files older than five years.

Archiving aging data

You can archive aging data without data deletion, file reorganization, or program changes. Once the oldest file is taken offline and archived, changing the group definition allows access to the remaining current files.

For example, if the group WEEK must always contain data from the most recent 7-day period, a new file is created each day and the oldest daily file is taken offline and archived. To preserve the files in chronological order, the group definition is modified to reflect the most recent seven daily files.

Accessing independent but similar data

You can access similar data kept in separate files either by the individual files or by a defined file group and an alternative OPEN command.

For example, if you keep data about individual states in separate files and you need reports for Ohio, a region, and the entire country, you can access the files dynamically using a file group. One user can select a report to be run against the OHIO file. Another user can generate a report for the region, OHIO, INDIANA, and ILLINOIS. Yet another user can generate a report for all 50 states.

Referencing new file names without changing application programs

You can use a file group to reference a new file without changing your application program:

1. Create a permanent group with the name referenced in the application.
2. Define the new file as the only member of the permanent group.
3. If this application is running as an application subsystem, make sure to define both the file and the permanent group in the subsystem definition.

For example, if several application programs operate on FILEA and FILEA is merged with FILEB under the name of FILEC, your application can access FILEC modifications by defining FILEC in a file group called FILEA. (This technique works if the applications use IN syntax rather than IN FILE syntax; see the *Rocket Model 204 User Language Manual* for more information.)

Updating File Groups

In a file group, when you create a new field, the earlier files in the group are unchanged. You can continue to query the database of all the files in the group. You do not need to add the new field to the earlier files. A field need only be present in one file in order to be referenced in a group context.

For example, in time series data, you might want to create a new field and preserve prior data unchanged. You can add a new file in the group containing the new field without affecting earlier files. When you initiate a query, all the files in database group are included in the search.

Creating the CCAGRP data set

Sizing CCAGRP

The size of CCAGRP depends on the maximum number of permanent groups used and the average number of files in a group.

A maximum of 61 group definitions can be stored on one page. This is the number of definitions that fit on a 6184-byte page if the definitions contain an

average of nine files. If the average group contains more than nine files, the number of definitions stored on each page is smaller.

The number of pages required is the sum of the number of definition pages, plus three pages for the File Control Table.

Space from deleted groups is reused.

z/OS procedures

In a z/OS environment, take the following steps to create CCAGRP:

1. Turn off the '2' bit of the SYSOPT parameter.
2. Insert a CCAGRP DD statement (which might have been previously allocated).
3. Insert a system manager login and password in the input stream after User 0's parameter line.
4. Insert the CREATEG command (no arguments) in the CCAIN stream.

The CREATEG run can also create permanent group definitions by issuing CREATE commands after the CREATEG command. For more information about CREATEG and CREATE, see the *Rocket Model 204 Parameter and Command Reference*.

5. Issue EOJ and bring up the run again with the '2' bit of SYSOPT turned on. CCAGRP is now accessible to the run.

The following JCL shows a sample batch run that creates CCAGRP.

z/OS JCL example

This sample batch run creates the CCAGRP data set:

```
//          EXEC PGM=BATCH204,REGION=350K,
//          PARM='SYSOPT=144'
//STEPLIB   DD DSN=M204.PROGMS,DISP=SHR
//CCASTAT   DD DSN=M204.CCASTAT,DISP=SHR
//CCAAUDIT  DD SYSOUT=A
//CCATEMP   DD UNIT=3330,SPACE=(TRK,5),
//          DISP=(NEW,DELETE)
//CCASNAP    DD SYSOUT=A
//SYSUDUMP   DD SYSOUT=A
//CCAGRP     DD DSN=M204.CCAGRP,UNIT=3330,
//          SPACE=(TRK,8),DISP=(NEW,CATLG)
//CCAPRINT   DD SYSOUT=A
//CCAIN      DD *
PAGESZ=6184
LOGIN USERID      ---- Log in as system manager
PASSWORD          ---- System manager password
```

```
CREATEG  
EOJ  
/*
```

z/VSE procedures

Before you can define permanent file groups to Model 204, you must allocate the CCAGRP file using the ALLOCATE utility (see Appendix C).

The Job Control Language statements required for the execution of the ALLOCATE program are:

```
// JOB ALLOCGRP ALLOCATE MODEL 204 PERMANENT GROUP FILE  
// DLBL M204LIB,'M204.PROD.LIBRARY'  
// EXTENT SYSnnn,...  
// LIBDEF PHASE.SEARCH=M204LIB.V411  
// DLBL CCAGRP,'permanent group file file-id',99/365,SD  
// EXTENT SYSnnn,balance of extent information  
// ASSGN SYSnnn,X'cuu'  
// EXEC ALLOCATE,SIZE=AUTO  
ALLOCATE FILE(CCAGRP)  
/*  
/ &
```

The following considerations apply to executing ONLINE, BATCH204, or user-written IFAM1 programs that use permanent group files:

- Provide the following label information in the execution job stream:

```
// DLBL CCAGRP,'permanent group file file-id',,DA  
// EXTENT SYSnnn,balance of extent information  
// ASSGN SYSnnn,X'cuu'
```

- Indicate the use of permanent group files via UPSI switch 6 (SYSOPT=2):

```
// UPSI xxxxxx1x
```

- Define the file type as DA.

z/VM procedures

In a z/VM environment, take the following steps to create CCAGRP:

1. Create a CCAIN file and name it CREATG CCAIN. Include the following statements in the file:

```
PAGESZ=6184  
LOGON SUPERKLUGE  
PIGFLOUR  
CREATEG  
LOGOFF
```

EOJ

2. Turn off the '2' bit of the SYSOPT parameter.
3. Create a CREATEG EXEC procedure (see the *Rocket Model 204 z/VM Installation Guide*).
4. Insert a FILEDEF for the CREATG CCAIN created above.
5. Create a CCAGRP file by executing the command:

```
ONLINE BYPASS CREATEG
```

Storing group definitions in CCAGRP

Permanent group definitions are stored in the Model 204 file CCAGRP, which must be created before any permanent groups can be defined.

All runs that access permanent file groups require:

- Statement in the JCL defining the CCAGRP file.
- '2' bit on in the setting of the SYSOPT parameter. (This setting lets Model 204 open CCAGRP to the run.)

Including CCAGRP in a Model 204 run does not require changes to the NFILES, NDCBS, or NDIR parameters normally specified. These parameters are automatically advanced by increments during system initialization to make room for CCAGRP.

CCAGRP is opened with UPDATE access during initialization if SYSOPT is set to OPEN CCAGRP. When a Model 204 job runs, it enqueues on CCAGRP in SHARE mode. If a command is entered to create or delete a permanent group, the enqueueing becomes EXCLUSIVE for the duration of the update. Thus, if more than one Model 204 system is running with CCAGRP, no permanent groups can be created or deleted.

Periodically back up permanent group definitions using the DUMPG and RESTOREG commands. The DUMPG and RESTOREG commands are described in the *Model 204 Parameter and Command Reference*.

You can delete permanent group definitions from the CCAGRP data set using the DELETE command. (See the *Rocket Model 204 Parameter and Command Reference*.)

Creating file groups

Use the CREATE command to create both permanent and temporary file groups. Only the system manager can create a permanent group.

Any user can create a temporary file group by entering:

```
CREATE [TEMP] GROUP groupname FROM filename [,filename  
•••]
```

```
[PARAMETER parameter list]  
.  
.  
.  
END
```

At least one file name is required. The following parameters can be included in the list. They are discussed in detail in the *Model 204 Parameter and Command Reference*.

Parameter	Description
UPDTFILE	Defines the file that receives stored records unless another file is specified by means of an IN FILE... STORE RECORD statement.
PROCFILE	Defines the group's procedure file.
BLDGFT	Creates a group's field table in memory.

Specifying PROCFILE = * creates a file group in which any or all files can contain procedures.

Examples

The following example shows the creation of a temporary file group:

```
CREATE GROUP REGION FROM DELAWARE, MARYLAND, -  
    VIRGINIA  
PARAMETER PROCFILE = VIRGINIA, BLDGFT = NO  
END
```

The following example creates a permanent file group with multiple procedure files:

```
CREATE PERM GROUP REGION FROM DELAWARE, MARYLAND, -  
    VIRGINIA  
PARAMETER PROCFILE = *, BLDGFT = NO  
END
```

Opening, closing, displaying, and deleting file groups

Opening and closing file groups

You open a file group by entering the OPEN or OPENC command.

OPEN Syntax {OPEN | OPENC} [PERM | TEMP] [GROUP] *groupname*

OPEN and OPENC operate identically, except that OPENC does not change the existing default. The default value TEMP applies when you have a permanent and temporary file group with the same name and do not specify which.

You can open several file groups during a single Model 204 session. They remain open until a CLOSE command is issued.

CLOSE Syntax `CLOSE [PERM | TEMP] [GROUP] groupname`

Closing a permanent file group closes each file in the group that was opened by a group OPEN. But it does not close files that were opened as individual files or as members of other groups. Closing a temporary file group does not close the files in the group.

Displaying file group information

The DISPLAY command displays information about one or more file groups.

DISPLAY syntax `DISPLAY [PERM | TEMP] GROUP`
 `[(display-option [, display-option ...])]`
 `{ ALL | groupname [, groupname ...] }`

If you enter:

```
DISPLAY GROUP ALL
```

the group name, status (permanent or temporary), member files, and group parameters are displayed for all the permanent groups and for your temporary groups. You can request that the command display only a list of group names and their status.

Deleting file groups

The DELETE command deletes both permanent and temporary file groups. Only the system manager can delete a permanent file group. Before you can issue the DELETE command for an open file group, you must close the group.

DELETE syntax `DELETE [PERM | TEMP] [GROUP] groupname`

File group passwords and privileges

File group passwords

Passwords are specified differently for permanent and temporary file groups. Opening a temporary file group is functionally equivalent to opening a series of files. As each file is opened, you are prompted for the file password, if one is required. The temporary file group is opened only after all the files have been opened.

Permanent file groups can have their own passwords. As with files, there are three types of permanent file groups:

- Public
- Semipublic
- Private

Passwords for semipublic and private file groups are supplied in the manner described for semipublic and private files.

File group privileges

Privileges for a temporary file group are derived from the privileges assigned to the files that make up the group. The privileges granted for a temporary file group are the most restrictive combination of member file privileges. (This combination is defined through a logical AND of the privileges of the individual files in the group.)

For permanent file groups, you can associate a different set of privileges with each group password. The privileges for a group determine which operations you can perform on the data and procedures stored in the group. You can establish default privileges for public and semipublic groups. The system manager enters all these group passwords into the system.

7

Creating Server Data Sets (CCASERVER)

In this chapter

- Overview
- Server swapping and the CCASERVER file
- Sizing and storage considerations
- Requirements for server swapping
- Allocation and job control

Overview

The CCASERVER file provides temporary space for storage of user work tables when the user has been moved (swapped) out of the server.

The system manager can enable server swapping simply by setting the NSERVS parameter to a value less than the value of NUSERS. Efficient use of swapping, however, requires careful attention to sizing and device-dependent considerations.

This chapter describes server swapping; discusses server sizing, disk storage requirements, and device dependencies; and explains how to allocate server data sets and implement swapping.

Server swapping and the CCASERVER file

Server swapping is a shared access method that the Model 204 ONLINE configuration can use to reduce server-area main-memory requirements. When a user is temporarily idle, Model 204 copies that user's tables and server data into a CCASERVER file. The server area is then freed to process another user.

Shared servers are divided into different sizes. An internal server pushdown list automatically assigns a user the smallest size that satisfies processing requirements. The LSERVPD parameter controls the size of the pushdown list.

The system manager can create as many as 101 server data sets in any operating environment. This can improve performance by spreading I/O activity across several disk devices. Naming and sizing rules for multiple server data sets are described in detail the section "Requirements for server swapping" on page 169.

CCASERVER in Storage feature

This feature is for sites which:

- Have z/OS in 64-bit mode and real storage larger than 2-gigabytes, or
- Run z/OS in 31-bit mode and have enough expanded storage to support dataspace with no excessive paging. A **dataspace** is an address space in z/OS with none of the control structures for tasks; it is simply data.

Other operating system configurations will begin excessive paging, and serious performance degradation of the entire system is likely to occur.

If you have enough real storage on your system, you can place either or both CCATEMP file and CCASERVER file in storage. The benefit to you of activating the CCASERVER in Storage feature depends on how much latent wait time is spent by your jobs waiting for the CCASERVER file I/O to complete.

- Users may server swap to CCASERVER file in-memory instead of to disk data set CCASERVER.
- There is no CYLINDER size restriction on SERVSZ, which is limited only by available memory.
- An additional benefit is that relief is provided on I/O subsystems. Because there is less contention, channels and disks are free to perform other services and jobs run faster.
- The APSYPAGE parameter may compliment the CCASERVER in Storage feature; see the *Rocket Model 204 Parameter and Command Reference* for details of the APSYPAGE parameter.

System programmers consideration

To use the CCASERVR in Storage feature, be aware that the maximum number of data- or hiperspaces and the total cumulative size of data- or hiperspaces for a given job may be controlled by the SMF IEFUSI exit.

In this exit, the parameter description shows that subwords 2 and 3 of the word 7 parameter are the keys, where:

- Word 7 subword 2 controls the maximum cumulative data- or hiperspace size
- Word 7 subword 3 controls the maximum number of data- or hiperspaces that a job may allocate

Check the IBM supplied settings and change them, if necessary.

Information regarding this exit routine is found in the *z/OS Installation Exits Document Number SC28-1753*.

Tracking dataspaces and hiperspaces

See the MONITOR DATASPACE command in the *Rocket Model 204 Parameter and Command Reference*.

Activating the CCASERVR in Storage feature for a job

To activate the CCASERVR in Storage feature, comment out the CCASERV DD statements in a job. Model 204 will allocate the correct amount of storage based on the SERVSZ and NSERVS parameters.

Sizing and storage considerations

Preparation for implementing server swapping involves calculating appropriate values for:

- SERVSZ parameter, which determines the largest server in the input stream
- Disk storage requirements, given the chosen value of SERVSZ and the device types available at your installation

For detailed instructions for estimating SERVSZ, see “Sizing user server areas” on page 46. This section explains sizing and storage calculations that have to do with the physical devices in use at your installation.

Physical devices and server size

The size of each server is likely to be limited by the physical device used for swapping.

When an FBA device is used, the server size is limited by the amount of virtual memory available to Model 204. All the servers defined for the run must fit, along with many other blocks of storage, in virtual memory.

When a CKD device is used, the server size is limited by the device's cylinder size (see Table 7-1).

Table 7-1. Server swapping constants for CKD devices

Device type	Device constant	Tracks per cylinder	Maximum server size
3380	47476	15	712140
3390	56664	15	849960
9345	46456	15	696840

Calculating CKD disk storage requirements

Use Table 7-1 to make the following calculation for CKD disk storage:

Tracks per server = $\text{SERVSIZE} / \text{Device constant}$ (rounded up)

Servers per cyl = $\text{Tracks per cyl} / \text{Tracks per server}$ (rounded down)

Cylinders = $\text{NUSERS} / \text{Servers per cyl}$ (rounded up)

Total Tracks required = $\text{Cylinders} * \text{tracks per cyl}$

Where

- *cyl* represents cylinder.
- *Device constant* is taken from Table 7-1 for the appropriate device type.
- *tracks per cylinder* is taken from Table 7-1.

Example 2 in the section “Disk storage calculation examples” on page 169 shows a CKD storage calculation.

Calculating FBA storage requirements

To calculate storage requirements for FBA devices, use the following formulas.

For FBA 3370 devices:

$n = \text{SERVSIZE} / 31744$ (rounded up)

Blocks per user = $n * 62$

FBA Blocks = $\text{NUSERS} * \text{Blocks per user}$

For FBA 9335 devices:

$n = \text{SERVSIZE} / 36352$ (rounded up)

Blocks per user = $n * 71$

$\text{FBA Blocks} = \text{NUSERS} * \text{Blocks per user}$

For FBA 9332 devices:

$n = \text{SERVSIZE} / 37376$ (rounded up)
 $\text{Blocks per user} = n * 73$
 $\text{FBA Blocks} = \text{NUSERS} * \text{Blocks per user}$

Disk storage calculation examples

1. The device is a CKD type (3380). The device constant from Table 7-1 is 47476, and there are 15 tracks per cylinder. SERVSIZE is 110000, and NUSERS is 9. The computation is:
2. The device is an FBA type (3370). SERVSIZE is 110000, and NUSERS is 9. The computation is:

$n = 110000 / 31744 = 3.46$ (round to 4)
 $\text{Blocks per user} = 4 * 62 = 248$
 $\text{FBA Blocks} = 9 * 248 = 2232$

Requirements for server swapping

Parameters

To implement server swapping:

- Set the number of servers (NSERVS) to a value less than the number of users (NUSERS).
If NSERVS is equal to NUSERS, or is allowed to default to the value of NUSERS, server swapping is not initiated.
- Set the value of the SERVSIZE parameter specified for the largest server in the CCAIN input stream to be at least as large as the aggregate table size for the largest user. To calculate the SERVSIZE parameter, refer to the discussion of server areas in the section “Server areas” on page 46.

Multiple server data sets

The system manager can specify up to 101 server data sets in any operating environment.

At z/OS and z/VM sites, server data set file names are CCASERVR, CCASERV0 through CCASERV9, and CCASRV10 through CCASRV99. At z/VSE sites, the file names are CCASRVR, CCASRV0 through CCASRV9, and CCASV10 through CCASV99.

The DD statements (or CMS FILEDEFS) are scanned first for CCASERVER, then for CCASERV0, CCASERV1, and so on. If a ddname is not found, then the system assumes that there are no more server data sets. For example, if there is no CCASERV4, Model 204 does not use CCASERV5, even if there is a DD statement for it.

Model 204 opens the data sets in consecutive order and makes allocations to users in round-robin fashion. As data sets fill up, they are dropped from the rotation.

Use multiple server data sets to spread the I/O activity across several disk devices. All the devices must be the same type (all FBA or all identical CKD devices).

Server data sets on CKD devices

If the server data sets are on CKD devices, they must begin on cylinder boundaries. Allocate each data set with contiguous space. The value of the SERVSZ parameter is limited to the cylinder capacity of the device containing the CCASERVER data set. A server is not allowed to span a cylinder boundary for performance reasons: each server read or write requires at most one seek.

ECKD channel program support

Model 204 executes ECKD channel programs for CCATEMP, CCAGRP, CCASYS, and CCASERVER on ECKD capable devices, whether the devices are connected to ESCON or parallel channels. All models of IBM 3990 Storage Control, the IBM 3880 Model 23 Storage Control, and all models of the IBM 9340 Direct Access Storage Subsystem support ECKD channel programs. This improves performance, particularly if you are using ESCON channels and/or IOS branch entry (XMEMOPT=2) under z/OS.

Under both z/OS and z/VSE, Model 204 executes ECKD channel programs for CCASERVER data sets on ECKD capable devices.

Under z/OS, I/O to user database files, CCATEMP, CCAGRP, and CCASYS is also done using ECKD channel programs.

Note: Under z/OS, VIO support is not available for use with EXCPVR or IOSB. Do not use 9345 DASD with EXCPVR because of the potential for poor performance on writes.

Under z/VSE, CKD channel programs are used to perform I/O to user database files, CCATEMP, CCAGRP, and CCASYS. I/O performance is not affected, however, because Model 204 uses a “well-formed” CKD channel program for these I/O activities. The z/VSE I/O supervisor efficiently converts the channel program to an ECKD channel program for ECKD capable devices.

Using cache fast write

If you have cached DASD controllers (such as IBM 3990 models 3 and 6) you can use the CACHE parameter to allow CCASERVER data to be written directly to a controller's cache.

Allocation and job control

z/OS

z/OS JCL must contain at least one DD statement for CCASERVER. You can specify a disposition of either SHR or OLD. SHR is recommended for the ONLINE configuration.

z/VSE

In a z/VSE environment, prior to executing the ONLINE configuration, you must initialize one or more server data sets using the ALLOCATE utility. For example:

```
// JOB ALLOCSRV ALLOCATE MODEL 204 SERVER DATASET
// DLBL M204LIB,'M204.PROD.LIBRARY'
// EXTENT SYSnnn,...
// LIBDEF PHASE.SEARCH=M204LIB.V210
// DLBL CCASVR,'MODEL204.SERVER.DATASET.1'
// EXTENT SYSnnn,balance of extent information
// ASSGN SYSnnn,X'cuu'
// EXEC ALLOCATE
ALLOCATE FILE (CCASVR)
/*
/ &
```

Label information must be inserted into the ONLINE job stream.

- For FBA devices:

```
// DLBL CCASVR,'MODEL204.SERVER.DATASET.1',,DA
// EXTENT SYS021,SYWK1,,,9000,1736
// DLBL CCASRV0,'MODEL204.SERVER.DATASET.2',,DA
// EXTENT SYS022,SYWK2,,,15032,496
```

- For CKD devices:

```
// DLBL CCASVR,'MODEL204.SERVER.DATASET.1',,DA
// EXTENT SYS021,SYWK1,,,1500,xx
```

where xx is a value from the Tracks per cylinder column in Table 7-1 on page 168.

FTBL and above the bar storage

FTBL can reside in 64-bit storage (above the bar) when Model 204 runs on z/OS or z/VM and storage above the bar is available.

You can allocate designated server tables for each user in storage above the bar that will not be subject to server swapping. This feature enables you to divide the server storage into two parts: swappable and non-swappable. This makes the swappable part of the server storage smaller and faster to swap.

Parameters

The SERVNSSZ and SERVNSA parameters control non-swappable server areas.

- SERVNSSZ (server non-swappable size) is the amount of space in bytes required for the above the bar server tables per user. This is a user 0 parameter applicable to all users. The total amount of storage allocated for non-swappable server parts equals SERVNSSZ rounded to 4 K and multiplied by NUSERS.

When sizing SERVNSSZ, use the largest FTBL sizes that might be needed.

- SERVNSA (server non-swappable areas) indicates the tables that you want to be above the bar. At this time, only FTBL may be selected and its appropriate setting is x'02000000'.

Description

The non-swappable server part may be used with server swapping done in storage or on disk. It may also be used when there is no server swapping (NUSERS=NSERVS) to make servers below the bar smaller by using the non-swappable server part above the bar and saving storage below the bar.

To obtain below-the-bar storage savings when the non-swappable server part is used, decrease the value of the SERVSZ parameter by the LFTBL value used in calculating the server size.

Note: When FTBL is placed in the non-swappable server part above the bar, then the UTABLE command decreasing FTBL size will not free any storage in the regular server below the bar.

Usage notes

You will need to ensure that your user-written and third-party \$functions can work with non-swappable server tables in 64-bit storage above the bar.

Messages

The following messages are associated with FTBL above the bar storage:

- M204.2916: SERVNSSZ REQUIRES AT LEAST ONE AREA TO BE SET IN SERVNSA
- M204.2920: Non zero SERVNSSZ is required
- M204.2921: Non swappable server areas are not supported in VSE
- M204.2922: SERVNSA bit setting is not valid
- M204.2923: Space required for non swappable server area is larger than SERVNSSZ

M204.2923 is issued when the space problem occurs during initialization. M204.2923 sets the return code to 80, appears in the job log, and results in the Online being immediately terminated.

- M204.2932: Space required for non swappable server area is larger than SERVNSSZ

M204.2932 is issued when the space problem is detected after initialization is complete, most likely during UTABLE command processing. It does not appear in the job log and does not result in the Online being terminated.

FTBL and above the bar storage

8

System Requirements for Application Subsystems

In this chapter

- Overview of CCASYS
- Maintaining CCASYS
- Using CCASYS
- Activating the APSY Precompiled Procedures In Storage feature
- Overview of the Subsystem Management facility
- Parallel Query Option/204 and scattered subsystems
- Components of a subsystem
- File and group availability
- Application subsystem processing
- Subsystem operating requirements
- Subsystem operating options
- Overview of the SUBSYSMGMT interface
- SUBSYSMGMT Activity screen
- Operational Parameters screen
- Procedure Specifications screen

- Subsystem File Use screen
- Subsystem Classes screen
- User Definitions screen
- Subsystem Class Users screen
- Subsystem Import/Export options
- Subsystem Administration screen
- Administrative Privileges screen
- Summary of subsystem privileges
- User Matrix screen

Overview of CCASYS

CCASYS is a system file used in conjunction with Dictionary and the Subsystem Management facility (SUBSYSMGMT). It contains the procedure names, file names, and parameters that create a subsystem definition. When a subsystem is executed, Model 204 opens and reads the CCASYS file.

The CCASYS file must be open to use the Subsystem Management facility and to run any user-written subsystem.

Statistics associated with CCASYS processing are written to the CCAUDIT system file and can be viewed from a terminal or printed to an audit trail.

This chapter describes CCASYS and explains how to create and maintain the file.

For details about the Model 204 Dictionary, refer to the *Model 204 DICTIONARY and Data Administration Guide*.

Maintaining CCASYS

Creating CCASYS

You can create CCASYS online as a separate file or as a part of the initial installation of Dictionary and the full-screen interface. Regardless of the approach used to create CCASYS, it must be available before Dictionary and the interface are installed.

A user with system manager privileges can create CCASYS online (see the CREATE command in the *Rocket Model 204 Parameter and Command Reference*). The formula for calculating the size of CCASYS is given in the Model 204 installation guides.

To create CCASYS during Dictionary installation, follow the instructions given in the Model 204 installation guide for your operating system.

Reorganizing CCASYS

For instructions on reorganizing CCASYS and Dictionary files, refer to the Dictionary section of the Model 204 installation guide.

Using CCASYS

You must be a system manager to run the Subsystem Management facility, as well as having privileges defined in the DICTIONARY to use SUBSYSTEMGMT. (Refer to the DICTADMIN subsystem to add privileges.)

To use the Subsystem Management facility and to run any user-written subsystem, the CCASYS file must be open. Model 204 opens CCASYS during initialization when the SYSOPT parameter includes a 1 (see *JCL requirements* on this page).

CCASYS is opened and locked in share mode. During the process of defining a subsystem, the lock is changed to exclusive mode for the duration of the session. If more than one copy of Model 204 is using the same CCASYS file, the Subsystem Management facility cannot be entered.

Subsystems available at initialization

If the CCASYS file is marked as full, FISTAT=X'08', during Online initialization, subsystems are now available and the following message is no longer issued:

```
M204.1457 UNABLE TO SCAN LIST OF SUBSYSTEM NAMES
```

JCL requirements

JCL requirements for CCASYS are:

- SYSOPT=1 value in the JCL PARM field to enable CCASYS.
 - SYSOPT=1 automatically increments NFILES, NDCBS, and NDIR to allow use of the subsystems.
 - If the SYSOPT parameter is set to an even value, CCASYS is not opened by Model 204 and subsystems are unavailable.
 - You must set the SYSOPT=X'01' bit on all nodes that the client subsystem accesses. (For Parallel Query Option/204)
- CCASYS DD statement for running the subsystem.

File security

You can add additional file security to the security provided by user privileges in the CCASYS file by:

- Resetting the OPENCTL parameter

- Entering passwords for the CCASYS file entries in the password table (CCASTAT). See “Using CCASTAT” on page 251.

Maintaining CCASYS

Periodically back up CCASYS. Like other Model 204 files, CCASYS can participate in recovery and transaction back out processing.

As necessary, include reorganizations in the recovery scheme, such as increasing both Table A and Table C and, sometimes, to recover a file if it is not participating in other recovery options. File reorganization is described in detail in the *Model 204 File Manager's Guide*.

Set up backup and restore jobs to handle all files at once. If need be, you can set up a separate backup job for CCASYS to run with the files. See the Rocket Model 204 installation guide for your operating system.

Recovering CCASYS

If CCASYS is used in a run, you can recover it with a RESTART command without resetting the SYSOPT parameter.

Synchronize recovery of CCASYS and Dictionary files. Dictionary subsystems and the Subsystem Management facility must be stopped to perform file maintenance.

Activating the APSY Precompiled Procedures In Storage feature

z/OS sites can keep precompiled procedures in storage. By removing precompiled procedures from the disk buffer pool, the buffer pool pages are available for Model 204 files or non-APSY CCATEMP pages. This can reduce disk I/O for both CCATEMP and other Model 204 files. The APSYPAGE parameter allows APSY saved compilations to be loaded without having the pages that the compilations are saved on move through the disk buffer pool. This represents a reduced load on page flushing and the HASH LOCK and reduced CPU usage.

To activate the APSY Precompiled Procedures In Storage feature in a job, users must set the APSYPAGE parameter in the USER0 CCAIN stream. The APSYPAGE parameter specifies the number of virtual storage pages to allocate.

Note: While APSYPAGE looks very much like TEMPPAGE, the units are somewhat different:

- APSYPAGE refers to 4096-byte virtual storage pages
- TEMPPAGE refers to 6184-byte Model 204 file pages

There are several reasons to use the APSYPAGE parameter instead of, or in addition to, the TEMPPAGE parameter:

- APSYPAGE virtual storage pages do not have to be big enough to hold all of CCATEMP or even all saved compilations. The APSY Precompiled Procedures In Storage feature uses an LRU algorithm to ensure that frequently loaded APSY procedures remain in storage, while infrequently loaded procedures tend to be loaded from CCATEMP. So, if there is not sufficient real storage to back all of CCATEMP, you might set APSYPAGE to save frequently loaded APSY compilations in real storage.
- The path length for retrieving a page out of storage is significantly lower than for retrieving one out of the disk buffer pool, especially in an MP/204 environment.
- When APSYPAGE is set, the large Model 204 server tables are (hardware) page-aligned as are the saved compilations. This makes it possible to use the hardware MVPG (MoVe PaGe) facility to move pages into a server during an APSY load. On some processors, specialized hardware makes the facility significantly faster than byte-oriented data movement.

Storage improvements for APSY subsystem saved precompilation

If not otherwise set, DSPOPT now defaults to X'00'. When a saved compilation page is moved to the APSYPAGE area, it is no longer also kept in CCATEMP, which eliminates duplicate storage.

This is particularly useful when you keep CCATEMP in memory, using the TEMPPAGE parameter. This reduces in memory CCATEMP storage requirements by approximately $(N/1.50)$, where N is the number of 4K-byte pages of APSYPAGE that are used. In memory CCATEMP size is defined by the TEMPPAGE parameter.

Setting DSPOPT=X'80' and TEMPPAGE greater than zero causes saved compilations to be saved in both CCATEMP and APSYPAGE. Although this is wasteful of storage, it does not cause problems. If you set both parameters, you may see a higher use of CCATEMP in storage.

Setting APSYPAGE greater than zero causes APSY subsystem precompiled pages to be saved in storage. If TEMPPAGE=0 is also set your CCATEMP is kept on disk and all saved compilations are stored in both locations. In memory APSYPAGE storage can still be reduced if DSPOPT is set to include the X'80' bit. With both those settings, Model 204 employs the least recently used (LRU) algorithm to keep the more heavily used saved compilations in memory while the less recently used saved compilations migrate to disk.

Storage requirements

The CCASERVER, CCATEMP and APSY Precompiled Procedures In Storage features cause operating system paging. Performance degradation is likely, if real storage frames are insufficient, or expanded storage frames in hiperspaces

are used to hold the data saved in this storage. The MONITOR DATASPACE command can provide some information about the storage usage of these features.

Setting the DSPOPT parameter

If the APSY Precompiled Procedures In Storage feature is used, or CCASERVER In Storage is used with page-oriented data movement (DSPOPT X'01' set), then certain large server tables and servers themselves are page-aligned. This might require up to five extra hardware pages of real and virtual storage per server, although more typically it requires about three extra pages or 12K bytes per server.

If NSERVS is set to 40 and APSYPAGE is set to a nonzero value, or CCASERVER is kept in storage with DSPOPT X'01' set, enough real storage frames should be available to hold up to an extra 200 pages or 800K of server data.

The setting of the DSPOPT parameter depends on your installation.

- If your site has enough expanded storage, you might use it for CCASERVER In Storage and APSY Precompiled Procedures In Storage in a cache hiperspace (DSPOPT=X'43').
- If this decision causes a large number of CCATEMP reads because z/OS steals a lot of pages from the cache hiperspace, then use DSPOPT=X'23' (no cache hiperspace).
- If there is an excessive paging, eliminate one or both In Storage features.
- If your installation is z/OS, 64-bit real-storage, do not use any hiperspaces options, but use dataspace instead.

Users with limited resources should try and test what feature is the most effective.

In addition, enough real storage should be available to hold the APSY precompiled procedures and/or CCASERVER data in real or expanded storage.

For more information about DSPOPT, see the *Rocket Model 204 Parameter and Command Reference*.

APSY load statistics

There are three system statistics associated with APSY loads. These statistics are intended to assist in the proper setting of the APSYPAGE and RESLTHR parameters and to determine the ratio of requests using precompiled and non-

precompiled requests, the percentage of requests which are precompiled being the ratio of APSYLD to REQ. The APSY load statistics are:

Statistic	Tracks the number of APSY loads...
APSYLD	Including internal CCASYS procedures that are run as part of APSY processing.
APSYLDD	From a dataspace. This statistic is zero unless the APSYPAGE parameter is set. This parameter can be compared with APSYLD and APSYLDT to determine the percentage of eligible APSY loads that are being performed from a dataspace.
APSYLDT	That are tiny. All APSY loads, even those done from dataspaces, read a certain amount of control information from CCATEMP. Immediately following this control information is data that is loaded into NTBL, QTBL, STBL, and VTBL. If this table data does not extend to an extra CCATEMP page beyond the control information, the APSY load is counted as a tiny load . This statistic is maintained because tiny APSY loads are not worth saving in dataspaces, so it must be considered when comparing APSYLDD with APSYLD in determining which percentage of pre-compiled procedures are being loaded from dataspaces.

Parallel Query Option/204 and scattered subsystems

Whenever you log in to a service subsystem, you must reset the LGTBL parameter on the service thread to at least 288. This sets the size of the global variable table (GTBL). GTBL contains name/value strings for global variables and is used by procedures that are included by APSY to support subsystem processing.

The subsystem designer will have to place the UTABLE command into the LOGIN procedure, if LGTBL is insufficient before entering the subsystem.

Overview of the Subsystem Management facility

The Subsystem Management facility (SUBSYSMGMT) is a full-screen Model 204 Dictionary interface. It is used as a tool to define user-written applications that run under the Application Subsystem facility (APSY).

Subsystem characteristics and components are defined by means of a series of screens and stored in the CCASYS file. Data in the CCASYS file is used by the application subsystem to control the processing required by the subsystem. Model 204 Dictionary entries are maintained for the operational parameters, procedure specifications, and subsystem files. The Subsystem Management facility makes User Language applications easier to use, easier to maintain, and more efficient.

The following sections provide an overview of subsystem processing and explains how to use the SUBSYSMTG interface to define a subsystem, establish user privileges, and migrate subsystem definitions from one Model 204 environment to another.

For more information on subsystems, requirements, and options, refer to:

- The Model 204 installation guides, which discuss the installation and preparation of CCASYS, D204SYS, and Dictionary files
- The *Model 204 DICTIONARY and Data Administration Guide*, which describes the dictionary entries relating to subsystems
- The *Parallel Query Option/204 User's Guide*, which describes a Model 204 distributed processing facility
- The *Model 204 User Language Manual*, which gives detailed instructions for designing, developing, and debugging a subsystem

Components of a subsystem

A **subsystem** is an application consisting of a collection of procedures, files, and assigned characteristics that are defined as a subsystem to Model 204 through the Model 204 Dictionary interface, SUBSYSMTG.

The collection of procedures that make up a subsystem perform system startup, login processing, main processing, disconnect processing, and error-handling tasks.

Control is passed from procedure to procedure by setting a defined communications global variable in each procedure. The communications global variable contains the name of the next procedure to be executed. Detailed information about subsystem design and procedure development is given in the *Model 204 User Language Manual*.

Assigned characteristics of a subsystem include security and system operation options, such as locking files and groups for subsystem use, automatic login and logout, automatic COMMIT for outstanding updates, message displays, and the response of the subsystem when files are unavailable.

The **members** of an APSY subsystem are files and permanent groups. With PQO, members can be either automatic or manual:

- An **automatic member** is a subsystem group or file that is opened automatically when the subsystem is started or when a user enters the subsystem.
- A **manual member** is a group or file that must be opened explicitly by the OPEN or OPENC command.

Members can also be either mandatory or optional:

- A **mandatory member** must be open in order to access a subsystem. Mandatory members cannot be manual.

- An **optional member** is not required for subsystem access (start and login processing can succeed without it).

At any given time, a member can be **open** or **closed** to a subsystem or to a user within a subsystem. The following sections explain the conditions under which the different kinds of members are accessible to APSY subsystems and their users.

File and group availability

Member availability to APSY subsystems

Automatic members of APSY subsystems are always opened by the START SUBSYSTEM command or by SUBSYSTEM LOGIN. At the end of START processing, each automatic member is open unless either the START or OPEN failed.

Manual members of APSY subsystems are in the closed state at the completion of START SUBSYSTEM processing and must be explicitly opened by the user. Manual members become open to the subsystem if an OPEN operation succeeds. If OPEN fails due to node unavailability or for user-specific reasons (for example, if the user's line goes down), the member remains closed to the subsystem.

- If a node becomes unavailable to a subsystem, all automatic subsystem members and all open manual subsystem members residing on the unavailable node are marked disabled.
- If a STOP FILE (or GROUP) command is issued for a manual member on the client subsystem's node, the member is closed to the client subsystem when the last user closes it.
- If the member is located on the service subsystem node, the file is closed to the service subsystem when either the STOP is complete or the last user closes the file.

Member availability to subsystem users

When a user enters a subsystem, automatic subsystem members are opened.

If a user LOGIN or OPEN operation fails for an optional member, the member is left closed for the user but remains open to the subsystem. If a mandatory member cannot be opened, the user is denied access to the subsystem.

If a user LOGIN or OPEN operation fails for an already open member, the member is left disabled for the user, but remains open to the subsystem.

If an automatic mandatory member is closed to the subsystem, new users are not allowed to enter the subsystem.

Manual members of APSY subsystems are closed for a user within a subsystem until the user issues an OPEN command or statement. In this case, it does not matter whether the member is open or closed to the subsystem.

If compilation and/or loading of a request fails due to a communications failure, previously opened members on the failing node become disabled to the user. A user can close optional members at any time by issuing the CLOSE command.

Enabling a disabled subsystem file

If a file is marked as disabled to the subsystem, you can use the ENABLE SUBSYSTEM FILE command to make the file available again. If necessary, correct any communications problem. Then enter this command:

Syntax `ENABLE SUBSYSTEM subsysname [FILE | GROUP] name`
 `AT location`

- Where**
- *subsysname* is the name of the client subsystem.
 - *name* specifies the file or group that supports *subsysname*.
 - *location* is the value of the client CCAIN LOCATION parameter (which is also the value of the CLNT field in the LOGWHO command output).

Note: You must specify the location; you cannot include local files in an ENABLE SUBSYSTEM command.

Disabling a subsystem file

You can disable a subsystem file to keep the file itself, or the subsystem to which it belongs, inaccessible for a short period of time, without shutting down the subsystem by using the DISABLE SUBSYSTEM command.

To disable a subsystem file, use this command:

Syntax `DISABLE SUBSYSTEM subsysname [FILE | GROUP] name`
 `AT location`

- Where**
- *subsysname* is the name of the client subsystem.
 - *name* specifies the file or group that supports *subsysname*.
 - *location* is the value of the client CCAIN LOCATION parameter (which is also the value of the CLNT field in the LOGWHO command output).

The consequence of disabling a subsystem file depends on whether the file is an optional or mandatory group member:

- If you disable a mandatory file, you effectively disable the subsystem, because users attempting to access the disabled file cannot access the subsystem.
- If you disable an optional file, users can log in to the subsystem, but cannot access the disabled file.

Application subsystem processing

Subsystem processing includes startup, login processing, locating, compiling and/or loading procedures (main or driver processing), disconnect processing, and error processing. User-written procedures are required for each processing step.

For startup and processing considerations specific to scattered subsystems, see the *Rocket Parallel Query Option/204 User's Guide*.

Subsystem startup

To start the subsystem, either issue the START command. Or, optionally, the first user starts the subsystem automatically, if the Auto Start option is in effect. During subsystem startup, Model 204 performs the following tasks:

1. Finds the subsystem definition stored in CCASYS.
2. Builds an in-core subsystem definition control block.
For z/OS operating systems, application subsystem control blocks reside above the 16-megabyte line.
3. Opens all required files and groups.
4. Scans the procedure file for locked members that have precompiled procedure prefixes (see "Procedure Specifications screen" on page 205).
5. Enters an entry for each precompiled procedure in the in-core procedure dictionary.

Note: The maximum size of the in-core procedure dictionary is 16 megabytes.

6. Adds the subsystem name to the list of active subsystems, if no error occurs.
7. Performs a user-written initialization procedure (optional).
8. Closes all associated files and groups, unless the Auto Start option is in effect.

Login processing

Each time the subsystem is invoked, Model 204 automatically performs the following steps:

1. Searches CCASYS for the user's class definition that assigns user privileges.
The subsystem invocation is rejected if privileges consistent with the subsystem invoked are not found.
2. Logs the user in to Model 204 with the subsystem name as the user, account, and record security ID, if the automatic login option (see "Operational Parameters screen" on page 202) is set in the subsystem definition.
If the automatic login option is not set, the user's Model 204 ID is retained.
3. Opens required subsystem files and groups with the found privileges.
4. Executes the commands and requests, as specified by the login procedure programmed by the user. The login procedure might consist of:
 - Storing current server table sizes in the global variable table for later reference
 - Issuing UTABLE commands to set compiler table sizes for subsystem use
 - Setting the communications global variable to the name of the procedure that displays the initial menu
5. Begins main processing as long as the login procedure specifies the next procedure to include in the communications global variable.

Main processing

Main processing consists of locating subsystem eligible procedures, compiling procedures (if necessary), and loading procedures until an exit value is encountered. You can have as many main processing procedures as necessary.

Procedure names are stored in the in-core procedure dictionary that is built during system startup. Procedures are retrieved by examining the communications global variable for the name of the procedure and then locating the procedure name in the in-core dictionary.

If the procedure name is not found, the subsystem error procedure is executed. If the procedure is not found and no error procedure is specified in the subsystem definition, the user is disconnected from the subsystem.

If the procedure name is found, Model 204 checks to see if the procedure has been precompiled:

- If the procedure is not a precompilable procedure, Model 204 includes it for compilation and evaluation. Compilation and evaluation are reported in the audit trail under the CMPL and EVAL since-last statistics.
- If the procedure is a precompilable procedure, Model 204 checks to see if the procedure was previously compiled with the set of privileges defined for the user's subsystem class.

- If the procedure is precompiled for the user's privilege set, Model 204 loads the contents of the compiler tables from CCATEMP and evaluates the request. Loading and evaluation are reported in the audit trail as LOAD and EVAL since-last statistics.
- If the procedure is not precompiled, Model 204 includes the procedure for compilation and evaluation. Contents of compiler tables are saved in CCATEMP, if the compile is successful and the tables are not already saved for another user class.
- If the procedure is already precompiled for some other privilege set, Model 204 validates whether this user's privilege set is authorized to compile and validate. If successful, the request is loaded and then evaluated. If unsuccessful (for example, if the privilege set fails), the error procedure is executed.

Main processing continues until the value of the communications global variable is set to the exit value specified in the subsystem definition. Once the exit value is set, Model 204 proceeds to disconnect processing.

Warning: Rocket advises that when using an INCLUDE statement in a precompiled procedure that the INCLUDE procedure is the same for all users. Unpredictable results may occur, if the procedure is different when compiling for different SCLASSES.

Requirements for using temporary groups

If a precompiled procedure includes an OPEN of a temporary group, an attempt to load that procedure fails (with the *GRP NOT OPEN* error) if the procedure contains an **unreferenced list** in the context of that group. An unreferenced list is a list that is declared but not referred to and, therefore, its declaration is not evaluated.

To avoid an error in this situation, make sure of the following:

- Any temporary group in this situation must be open at the time of the load.
- Such a temporary group must be a TBO group if the procedure accesses a TBO file for update.

Disconnect processing

Disconnect processing is invoked when one of the following conditions occurs:

- Communications global variable is set to the exit value.
- Error occurs with no subsystem error procedure defined.
- User is restarted by Model 204.
- Logout or disconnect commands are issued from a subsystem procedure.

All open subsystem files and groups are closed for the user during disconnect processing. If the subsystem definition includes the automatic logout option (Log User out of M204 on the Operational Parameters screen), the user is logged out of Model 204.

If the subsystem definition includes the automatic login option (Log User in to M204) but not the automatic logout option (Log User out of M204), the user is exited out of the subsystem, logged in again under the original Model 204 ID (if previously logged in), and returned to Model 204 command level.

Error processing

Error processing, an optional procedure, is invoked when an error occurs that cannot be handled by the procedure executing at the time.

Recoverable errors

A subsystem can recover from most errors when an error procedure is used. Recoverable errors include:

- Compilation errors
- Record locking or table-full errors
- Attention interrupts and *CANCEL if no ON ATTENTION unit is specified in the application

An error procedure must test for different error conditions. The resulting value stored in the error global variable helps the application programmer determine the type of error that occurred.

Considerations for the communications global variable

The communications global variable is ignored and disconnect processing completed when one of the following conditions occurs:

- Error is a soft restart, a hard restart, or a phone hang-up condition.
If you attempt to set the communications global variable to the name of another procedure, the procedure is not executed.
- No error procedure is specified in the subsystem definition.

Subsystem operating requirements

Meet the following requirements to define and execute a subsystem:

- Install CCASYS, the Model 204 Dictionary, and the subsystem interface SUBSYSMGMT
- Allocate sufficient space for subsystem use in the resource locking table, server tables, CCATEMP, and spare core (SPCORE)

Required files

Dictionary/204, with the files shown in Table 8-1, must be installed to define a subsystem; see the Rocket Model 204 installation guides. Dictionary entries pertaining to subsystems are for reference purposes only. The subsystem is executed by using the data stored in CCASYS.

Table 8-1. Dictionary/204 files required for subsystem management

File	Description
CCASYS	Subsystem definition
D204SYS	Migrating definitions
DATALINK	Dictionary file
M204PROC	Procedure file
M204TEMP	Internal work file
METADATA	Dictionary file

Resource locking table space

Model 204 locks in share mode on subsystem procedure names or permanent group names. Locking ensures that the procedures or group definitions do not change while the subsystem is running and prevents any user from issuing the PROCEDURE, DELETE PROCEDURE, or RENAME PROCEDURE commands. Procedures cannot be updated with the editor while the subsystem is active.

Separate locks are not required if the subsystem definition specifies locked files during processing.

Additional space might be required in the resource-locking table for running a subsystem with groups defined or files unlocked. Use the following formula as a guide:

```
Number of additional resource-locking entries =
Number of groups + number of procedures
                    (if files are unlocked)
```

where additional resources include:

- Seven additional entries for application subsystem procedures in CCASYS
- One additional entry for each subsystem and permanent group
- One additional entry for each subsystem procedure if LOCK FILE (or GROUP) options are chosen

Minimum server table sizes

Minimum server table lengths, exclusive of any application procedures, for all users invoking subsystems are:

- LGTBL = 288
- LNTBL = 50
- LQTBL = 120
- LSTBL = 250
- LVTBL = 256

During application programming, you must determine actual table lengths for specific procedures. Make adjustments to table lengths in the login procedure to ensure successful execution.

CCATEMP space

CCATEMP requires additional space to accommodate the compiler tables for precompiled procedures. The data stored in CCATEMP consists of a header section and the contents of GTBL, NTBL, QTBL, STBL, and VTBL.

Use the following calculation to determine the additional CCATEMP space required for one precompiled request:

```

94 + (32 * VTBL HWM) + (12 * NTBL HWM) + (STBL HWM)
    + NFILES + NRMTFILE + (16 * QTBL HWM)
    + (ad hoc group FTBL space)
    + ((30 + (7 + NRMTLOCS)/8) * #groups)
    + (8 * (#fields referenced in group))
    + (the sum of the length of the names of all the
      fields referenced in the group)
    + #screens + #images
    + (unavailable-file space) + (XVAR space)

```

where:

- *HWM* refers to the high water mark found in the audit trail's since-last compilation statistic for the indicated table.
- *ad hoc group FTBL space* depends on whether ad hoc scattered groups (PQO) are included. If no ad hoc groups are scattered, ad hoc group FTBL space is:

```
62 * (#ad hoc groups)
```

If some ad hoc groups are scattered, *ad hoc group FTBL space* is:

```
(62 + (#open files in ad hoc groups)) * (#ad hoc
```


groups)

- *unavailable-file space* is the following quantity, not knowable in advance, which you need to estimate (PQO only):

$$(4 + (\#unavailable\ group\ files)) * (\#groups\ with\ unavailable\ members)$$

- XVAR space is required for these User Language request elements: found sets, lists, and FOR statements with a WHERE clause. The number of bytes per element depends on the file or group context, as follows:

Transaction context	Bytes required per element
Single file	8
Ad hoc or permanent group	8 * (#files in group)
Temporary group	4 + (8 * (#files in group))

The number of additional CCATEMP pages required is:

The result of the above calculation / (PAGESZ - 40)

SPCORE size

Control blocks are allocated from spare core for use by active subsystems. Use the following calculation to determine the number of bytes required for one subsystem. The calculation includes PQO remote file subsystem members:

$$\begin{aligned} &228 + NRMTFILE + NFILES + (40 * SCLASSes) + NRMTLOCS \\ &+ (93 * filemembers) \\ &+ ((36 + (7 + SCLASSes) / 8) * (procedures + 1)) \\ &+ 40\ bytes\ for\ each\ stopped\ file\ or\ group \end{aligned}$$

where:

- *NRMTFILE* and *NRMTLOCS* are CCAIN parameters that apply to Parallel Query Option/204.
- *filemembers* are the files that belong to the subsystem.
- *SCLASSes* is the number of subsystem user privilege classes.
- *procedures* is the number of precompiled procedures used by the subsystem.
- *40 bytes for each stopped file or group* is the number of bytes returned to spare core if the subsystem is stopped or the file or group is started.

Subsystem operating options

Subsystem operating options consist of commands entered on the command line, the AUTOSYS parameter entered on user lines in CCAIN, and values

entered on screens provided by the Application Subsystem. Options can be entered on the command line only if the subsystem is already started or if the subsystem is defined with the Auto Start option. A summary of the available options is given in Table 8-2.

More detailed information about the operating options follows the table. Options entered through interface screens are discussed throughout this chapter.

Table 8-2. Summary of subsystem operating options

Option	Method of entry	Purpose
Auto Commit	Operational Parm screen	Controls automatic COMMITs at the END of each procedure
Auto Start	Operational Parm screen	Automatically starts the subsystem when the first user logs in
Automatic Login (Log user in...)	Operational Parm screen	Logs the user in to the subsystem with an account value of the subsystem name
Automatic Logout (Log user out...)	Operational Parm screen	Controls user logout from Model 204
AUTOSYS	CCAIN user line	Invokes the specified subsystem for individual users upon Model 204 login
Commands		
DEBUG SUBSYSTEM	Command line	Allows the user issuing the command to change subsystem procedures without stopping and restarting the subsystem
DISABLE SUBSYSTEM FILE	Command line	Makes a file or subsystem temporarily inaccessible
ENABLE SUBSYSTEM FILE	Command line	Makes a file available again after being disabled
START FILE	Command line	Allows a file to be opened and removes the stop flag (if any)
START SUBSYSTEM	Command line	Activates the subsystem, opens files, and performs subsystem startup
STOP FILE	Command line	Prevents opening a specified file or permanent group
STOP SUBSYSTEM	Command line	Stops the subsystem
TEST	Command line	Creates a single user test environment

Table 8-2. Summary of subsystem operating options (Continued)

Option	Method of entry	Purpose
File/Group	Command line	Specifies the name and attributes of files and groups used by the subsystem
Global variables		
Communication	Procedure Spec screen	Specifies the variable name of the next procedure to be executed
Exit Value	Procedure Spec screen	Specifies the communication variable setting for exiting the subsystem
Error Variable	Procedure Spec screen	Names the variable containing the error code
Iterations	Operational Parm screen	Specifies the maximum number of times a procedure can execute consecutively
Lock File/Groups	Operational Parm screen	Controls outside user access to subsystem files when the subsystem is active
Message display		
Disconnect	Operational Parm screen	Controls disconnect message display
Informational	Operational Parm screen	Controls the display of informational messages
Error	Operational Parm screen	Controls error message display
Numlk	File Use screen	Specifies the number of files participating in procedure locking
Proc names		
Initialization	Procedure Spec screen	Names the first procedure used upon subsystem startup
Login	Procedure Spec screen	Names the first procedure executed for each user
Error	Procedure Spec screen	Names the procedure invoked when an error occurs
Proc prefixes		
Non-precompiled	Procedure Spec screen	Identifies the prefix used for procedures compiled each time they are invoked

Table 8-2. Summary of subsystem operating options (Continued)

Option	Method of entry	Purpose
Precompiled	Procedure Spec screen	Identifies the prefix used for procedures that are saved in CCATEMP for reuse
Procs	File Use screen	Names the file or group containing the subsystem procedures
Security		
Account	Operational Parm screen	Specifies an account value that overrides the login account
Account	Subsys Class screen	Specifies an account associated with a specific user class (overrides the account specified on the Operational Parameters screen)
Account	User Matrix screen	Changes an account in any subsystem user class
Command Priv.	Subsys Class screen	Specifies whether the class of users can issue the START, TEST, STOP, and DEBUG subsystem commands
File Priv.	Subsys Class screen	Specifies values for procedure, file, and field-level security parameters
Login Priv.	Subsys Class screen	Overrides privileges on entry to the subsystem and privileges specified on the Operational Parameters screen
Privileges	Operational Parm screen	Specifies user privileges independent of the Model 204 login privileges
Record Security	Subsys Class screen	Overrides the record security ID held upon entry into the subsystem
Start Login	Operational Parm screen	Controls login privileges while starting a subsystem
Status	Operational Parm screen	Specifies the level of availability of the subsystem to users
Sys Class	Subsys Class screen	Specifies the number and name of a subsystem user class that is assigned specific privileges
Subsys Class	Subsys Class Users screen	Specifies subsystem class changes for individual users
Subsys Class	Userdef screen	Identifies a group of subsystem users and specifies the command privilege level applicable to that group

Subsystem commands

The following list summarizes commands relating to a subsystem. For more information, see the *Rocket Model 204 Parameter and Command Reference*.

- *DEBUG SUBSYSTEM* allows login to a specified subsystem with the *DEBUG* and *STATS* options of the *TEST* command. Individual programmer changes to subsystem procedures can be made without stopping and restarting the subsystem. Changes are limited to the programmer issuing the *DEBUG SUBSYSTEM* command. More than one user can execute the *DEBUG* command against the same subsystem at the same time.

DEBUG SUBSYSTEM displays the value of the communications global variable and since-last statistics. Prompts are issued for changes to the variable.

DEBUG is limited to users with either *TEST* or *DEBUG* privileges. The *DEBUG* privilege does not entitle use of the *TEST* command.

- *DISABLE SUBSYSTEM FILE* makes a file or the subsystem to which it belongs inaccessible for a short period of time without shutting down the subsystem by using the *STOP SUBSYSTEM* command.
- *ENABLE SUBSYSTEM FILE* makes a file marked as disabled to the subsystem available again.
- *START FILE* removes the *OPEN* restriction set by the *STOP* command at the system and subsystem levels.
- *START SUBSYSTEM* activates the subsystem and makes it available for use. Model 204 opens all the files and performs subsystem startup.
- *STOP FILE* prevents the reopening of a specified file or permanent group. Files or groups open at the time *STOP FILE* is issued are not affected until they are closed and an attempt is made to reopen them.

When the stopped file is closed by all users, all precompiled code in *CCATEMP* that refers to the file is discarded and pages in the temporary file are reclaimed. *STOP FILE* affects only one copy of Model 204. Another copy of Model 204 can process the file or group.

STOP FILE verifies the presence of the named file or group in any active subsystem. Files or groups not required by the subsystem become unavailable for future use.

- *STOP SUBSYSTEM* stops the subsystem and makes it unavailable for use. Files and groups are closed. Groups and procedures are released when the subsystem is completely stopped.

If *STOP SUBSYSTEM* is issued when there are active users, the system remains active until the last user disconnects (drain state).

- *TEST* creates a single user test environment. The subsystem must be stopped to enter the *TEST* mode and the user must have *TEST* privileges. *DEBUG* privileges are not sufficient to execute the *TEST* command.

TEST without the DEBUG option simulates execution of the subsystem.

AUTOSYS parameter

The AUTOSYS parameter, specified on user lines in CCAIN, invokes a subsystem for individual users upon login to Model 204. The same subsystem is generated for each subsequent user line until a different subsystem name is specified, or AUTOSYS is set to C' '.

In the following example, all IUCV users enter the subsystem VMCFPROF whenever they log in. The AUTOSYS parameter is turned off for all other users.

```

IODEV=41 , POLLNO=1 , NOTERM=50 , AUTOSYS=C' VMCFPROF'
IODEV=41 , POLLNO=2
IODEV=41 , POLLNO=3
.
.
.
IODEV=43 , AUTOSYS=C' '
IODEV=43
IODEV=43

```

Overview of the SUBSYSMGMT interface

SUBSYSMGMT is the full-screen interface used to define user-written applications that run under the Application Subsystem facility.

To issue a command from the SUBSYSMGMT screens, press the assigned PF key or enter the command on the command line (==>). Abbreviations for commands are indicated by the uppercase portion of the command listed for PF keys. Table 8-3 describes common commands and PF keys for SUBSYSMGMT screens.

Table 8-3. Commands common to SUBSYSMGMT screens

Command	PF key	Purpose
HELP	PF1	Displays online HELP text for the screen.
QUIT	PF3	Terminates processing without saving screen input. QUIT returns to the previous level.
END	PF12	Saves input and returns to the previous level.
	PF11	Generally used to move to the next screen when adding, modifying, or browsing a subsystem. The command name depends upon the name of the screen.

Error messages are issued from all screens. Messages can be general and issued from any screen, or specific and issued from a particular screen.

SUBSYSMTG facility screen summary

Table 8-4 summarizes the SUBSYSMTG facility screens. The screens are described in detail in the sections that follow.

Table 8-4. SUBSYSMTG facility screens

Screen	Purpose
Primary screen	
Activity	Provides a menu from which to select subsystem definition activities
Secondary screens	
Operational Parameters	Defines the subsystem operating parameters
Procedure Specifications	Defines subsystem procedure specifications
Subsystem File Use	Defines file names used by the subsystem
Subsystem Classes	Defines command and file privileges for each class of subsystem user
User Definitions	Defines the users of the subsystem when used in conjunction with the Subsystem Class User screen
Subsystem Class Users	Defines assignments of specific user accounts to specific user classes
User Matrix	Changes the definition of a single account in a user class
Subsystem Administration	Defines user and administrative privileges
Subsystem Trust	(PQO only) Views or manages trusted subsystem definitions

Using secondary screens

The following considerations apply to all secondary screens:

- Use secondary screens in any order.
- Use PF keys to move from screen to screen.
- You must fill in all screens, except those for public and semi-public subsystems with only one class, to complete a subsystem definition.
- Public and semi-public subsystems with only one class do not require the use of the User Definitions (Userdef) screen.
- The END command (PF12) stops and saves the definition when it is complete. If you use the END command with an incomplete definition, you receive a warning message. Issuing a second END command allows you

to exit and you can complete the definition in another session. (Do not use an incompletely defined subsystem.)

Sample screens and PQO

For screens that have changed since the previous release, the sample screens shown in the following sections represent definitions for a client subsystem in a PQO application.

PQO-specific fields are explained briefly but their use is not described in detail. Not shown here is the Subsystem Trust screen, which is specific to PQO.

For full explanations of client and server definitions in PQO applications, see the *Rocket Parallel Query Option/204 User's Guide*.

SUBSYSMGMT Activity screen

The Activity screen is the primary SUBSYSMGMT screen. Use it to access all SUBSYSMGMT functions.

```

SUBSYSMGMT                      Subsystem Management Facility                      VER 7 REL

                                2 Select Subsystem Activity
                                -----
                                1. Add
                                2. Modify
                                3. Browse
                                4. Copy
                                5. Rename
                                6. Delete
                                7. Import
                                8. Export
                                9. Export Delete
                                10. ADMIN

Subsystem Name: sub1           From:
Copy/Rename To:               From:
Export Users ?: N

===>

1=HELP      2=FILEuse  3=QUIT    4=OPERation  5=PROCEDURE  6=SYSclass
7=CMDprv    8=        9=USERdef  10=TRUst    11=EXPOrtlist 12=

```

Figure 8-1. Subsystem Activity screen

Commands and options

PF2, PF4, PF5, PF6, PF7, or PF9 are used with ADD, MODIFY, or BROWSE to specify the appropriate secondary screen. The Exportlist option (PF11) is described in the section “Exportlist” on page 218.

Overview of activities

The following options are initiated from the Activity screen and completed through the selected secondary screens:

- ADD a new subsystem definition
- MODIFY an existing subsystem definition
- BROWSE without updating an existing subsystem definition

The following options are initiated and completed from the Activity screen:

- COPY an existing subsystem definition to a newly named subsystem definition
- RENAME an existing subsystem with a new, unique name
- DELETE an entire existing subsystem definition
- IMPORT subsystem definitions
- EXPORT subsystem definitions
- EXPORT DELETE a subsystem from the D204SYS file

The ADMIN option (option 10) displays secondary screens on which the system manager can define privileges and assign and copy SCLASS membership.

From fields (PQO only)

The From fields (used when defining a PQO service subsystem) specify the location of the client subsystem. If you specify a From field, you cannot leave the Subsystem Name field blank.

The From field is prefilled on all the secondary screens and protected from input.

Revising command privileges

You can revise command privileges, if you have privileges to use the SUBSYSMGMT facility as well as the privileges to update the individual subsystems.

Using the main screen of the SUBSYSMGMT facility you can enter a pattern in the Subsystem Name field and select PFkey 7 to change the command privileges for a set of subsystems. Another screen appears where you can enter a pattern for a subsystem class along with the new command privileges for the START, STOP, TEST, DEBUG, RESUME, SUSPEND, REFRESH commands. You can also obtain a list of all of the subsystem classes and subsystem definitions that fit the pattern criteria.

To update the command privileges, select the 2. MODIFY option on the main menu, enter a subsystem name or pattern in the Subsystem Name field and press PF7. The COMMAND PRIVILEGES screen is displayed next. However:

- If you enter a pattern in the name field, but press another function key, the pattern is treated as an individual name, not a pattern. PF7 is only valid with the MODIFY option.
- If PF7 is pressed, but the MODIFY option was not specified, then the following error message is displayed:

SUMnnn: PFkey or command only valid with Modify option.

- If the Subsystem Name is left blank, the following error message is displayed:

SUMnnn: Subsystem name or pattern is required.

SUBSYSMGMT Privileges screen

The subsystem management command privileges screen is shown below:

SUBSYSMGMT		COMMAND PRIVILEGES						
< 11 > of < 20 >		----- Change all selected: -----						
Subsystem	Class	Start (Y/N)	Stop (Y/N)	Test (Y/N)	Debug (Y/N)	Resume (Y/N)	Suspend (Y/N)	Refresh (Y/N)
SUB1	*							
x SUB1	SC19	N	N	N	N	N	N	N
x SUB1	SC2	N	N	N	N	N	N	N
x SUB1	SC3	N	N	N	N	N	N	N
x SUB1	SC4	N	N	N	N	N	N	N
x SUB1	SC5	N	N	N	N	N	N	N
x SUB1	SC6	N	N	N	N	N	N	N
x SUB1	SC7	N	N	N	N	N	N	N
x SUB1	SC8	N	N	N	N	N	N	N
x SUB1	SC9	N	N	N	N	N	N	N
x SUB1	USERS	N	N	N	N	N	N	N
===>								
1=HELp	2=	3=QUIt	4=LIST	5=DESelect all	6=			
7=BACKward	8=FORWARD	9=	10=	11=UPDate priv	12=END			

Figure 8-2. Subsystem Command Privileges screen

After your screen entries are read, a list of subsystem names and subsystem classes based on the pattern criteria is displayed. The pattern criteria used for SUBSYSTEM NAME is the value that was specified on the main screen. The pattern criteria for SUBSYSTEM CLASS defaults to '*' when the screen is initially read.

If any of the subsystems in the list are enqueued by other users then the command privileges for the subsystem cannot be updated. In this case, 6=DISplay appears on the screen to display the list of subsystems that are enqueued.

Choosing other classes or names

If you wish to refine the list of Subsystem Classes or Subsystem Names then you may enter a new pattern in the Subsystem Name or Subsystem Class field and press PF4 to view the new list.

The maximum number of subsystem classes that may be processed at one time is 1000. If the list exceeds 1000, then the following error message is displayed:

```
SUM096: Refine criteria: # of subsystems classes, exceeds
max(1000) .
```

Entering changes in the COMMAND PRIVILEGES screen

To update the command privileges for a particular subsystem class the command privilege must be set to 'Y' or 'N' in one or more of the subsystem command privilege fields. In addition, the Subsystem Class must be selected by placing an 'x' in front of the subsystem name and the class name. You can also update more than one subsystem class at a time by specifying a command privilege of one of the 'Change All Selected' columns. If the Subsystem name and class are not selected then the command privileges are not updated for that particular subsystem class.

At this point, the user presses PF11/UPDate or PF12/END to update the command privileges for the entire set of selected subsystem classes. If a record enqueueing conflict occurs during the up[date process then only a partial update can be performed. If this should occur then the user can press PF6/DISplay to display the list of subsystem classes that are enqueued and cannot be updated at this time. The user also has the option of pressing PF11/UPDate or PF12/END a second time, to apply the partial update or press PF3/QUIT to not apply the update.

Command privileges are in effect immediately after they are updated and can be changed if the subsystem is active or not.

You may also deselect particular subsystems or classes on the list by deleting the X that precedes the subsystem or class name.

On Table 8-5, the list of PF key options and commands are alternatives that accomplish the same thing. (Commands are used with the Enter key.) Each PF key function listed can also be performed by adding 12 to that PF key and using the resulting PF key. Commands shown may be abbreviated to the first three characters, which are capitalized.

The procedure performs one of the following operations based on the PF key pressed, as listed in Table on page 202.

Table 8-5. PF keys

PF Key	Command	Purpose
1	HELp	Display help information on this screen
3	QUIT	Exit from Command Privileges Screen return to main menu for SUBSYSMT.
4	LIST	Displays a list of subsystems and classes based on the pattern specification in the Subsystem Name and Class fields.
5	DESelect all SElect all	De-select all subsystem names on the list. Select all subsystem names on the list. Toggles between DESelect and SElect.
6	DISplay	Displays a list of subsystems or classes that are enqueued by another user. The command privileges for the subsystems and classes on the list cannot be updated at this time.
7	BACKward	Scroll up on subsystem names list
8	FORward	Scroll down on subsystem names list
11	UPDate	Update all subsystem classes with new command privilege. Only the command privileges that are changed are updated.
12	END	Return to the main menu after updating the command privileges that are specified. If a particular command privilege is left blank then it is not updated.

Command privileges can be changed regardless of whether a subsystem is active or not. The new privileges are in effect immediately once they are changed.

Operational Parameters screen

Selecting PF4 from the Activity screen brings up the Operational Parameters screen. Use it to specify the operating parameters of the subsystem.

On this screen, only system managers can modify Account, Privileges, and Login Privileges fields.

```

SUBSYSMGMT                      Operational Parameters
                                Update Mode      From:
Subsystem Name: SALES
Enter Status: 1 ( 1. PUBLIC   2. SEMI-PUBLIC   3. PRIVATE )
Auto Start: N
Lock File/Groups: N
Log user into M204: N
Log user out of M204: N
Auto Commit: Y
Maximum Iterations: 5
Account: SALEACCT
Privileges (in HEX):
Start Login privileges (in HEX):
Subsystem can access Remote Files: Y

Message Display Options
Disconnect: Y
Informational: Y
Error: Y
===>

1=HElp      2=FILEuse    3=QUIT    4=          5=          6=SYSclass
7=          8=          9=USERdef  10=         11=PROCEDURE 12=END

LU008 PLU

```

Figure 8-3. Operational Parameters screen

Commands

- PF2, PF6, PF9 and PF11 validate and save input and display other secondary screens. You must use a PF key to quit, save the input, or move to a different screen.
- ENTER validates screen input.

A description of each parameter follows.

Parameter descriptions

- **STATUS** (1, 2, and 3) determines access availability:
 - Public (default) allows any user to access the subsystem.
 - Semi-public allows all users to access the subsystem, but permits different privileges for each class of users.

A semi-public subsystem generally has more than one user class. One class is designated as the default (see “Subsystem Classes screen” on page 210).

 - Private requires all users to have an assigned class. No default is allowed.
 - **AUTO START** automatically starts the subsystem when the first user enters the subsystem name. The START command is not required.
- If N (the default) is specified, only a user having the appropriate privileges, as defined on the Subsystem Classes screen, can issue the START command to open the subsystem.

- *LOCK FILE/GROUPS* provides control for file access to users outside the subsystem.

If Y is specified, the subsystem files and groups are available only to users running in the subsystem after the subsystem is started.

A subsystem file can be opened by a non-subsystem user if the subsystem is not started. If any user has the file open when the subsystem is being started, the subsystem start fails.

If a group is defined, all member files of the group are locked. No explicit lock is held on the group.

If N (default) is specified, users outside the subsystem can open any subsystem file, but cannot DELETE, RENAME, or REDEFINE fields.

- *LOG USER INTO M204* controls the method of logging the user in to Model 204.

If Y is specified, the user is logged out of Model 204 and then logged back in to Model 204 with the subsystem name as the user ID when entering the system.

If N (default) is specified, the user remains logged in under the same Model 204 account name used before subsystem processing.

- *LOG USER OUT OF M204* controls the method of logging the user out of Model 204.

If Y is specified, the user is logged out of Model 204 upon leaving the subsystem.

If N (default) is specified and LOG USER INTO M204 = Y, the user's logon, account, and record security ID are restored to the values that were present before entering the subsystem.

- *AUTO COMMIT* controls the issuing of COMMIT statements.

If Y (default) is specified, a User Language COMMIT statement is executed by Model 204 at each procedure END in the subsystem during execution.

If N is specified, transactions can span request boundaries (see "Transaction boundaries" on page 352). The application must issue COMMIT statements to commit updates to the database.

- *MAXIMUM ITERATIONS* specifies the maximum number of consecutive times the subsystem allows the same procedure to be invoked before interrupting the processing. Valid values are 1 to 99999. NULL (default) indicates no limit.

- *ACCOUNT* (optional) specifies an account value other than that used at logon. Up to 10 characters can be entered. NULL can be used to specify the logon account value.

The original value is restored when the user exits the subsystem.

- *PRIVILEGES* (optional) specifies a user's privileges while in the subsystem. Privileges specified before logging into the subsystem are

overridden, regardless of the value of the LOG USER INTO M204 field. The original value is restored when the user exits the subsystem.

The privilege option is expressed as a hexadecimal value within the range of X'00' to X'FF', as described in "File security" on page 270.

- *START LOGIN PRIVILEGES* specifies user login privileges for use while running the subsystem initialization procedure. If specified, START LOGIN PRIVILEGES overrides both the user's previous privileges and other privilege fields in the subsystem definition.

In the case of automatic login, the starting user's privileges are reset to those set on the Operational Parameters or Subsystem Classes screens prior to continuing execution within the subsystem.

The privilege option is expressed as a hexadecimal value within the range of X'00' to X'FF', as described in "File security" on page 270. NULL indicates that other privilege specifications are not to be overridden.

- *SUBSYSTEM CAN ACCESS REMOTE FILES* is a new field pertaining to PQO applications. N is the default. Change the value to Y, if you want the subsystem to access remote files.

Message display options

At the bottom of the screen are the message display options:

- *DISCONNECT* controls the display of subsystem disconnect messages:
 - If you specify Y (default), the subsystem disconnect message is displayed.
 - If you specify N, the subsystem disconnect message is suppressed.
- *INFORMATIONAL* controls the display of informational messages.
 - If you specify Y (default), Model 204 informational messages are displayed.
 - If you specify N, Model 204 informational messages are suppressed on the user's terminal and only subsystem messages are displayed. Model 204 messages continue to be printed on the audit trail.
- *ERROR* controls the display of error messages on the user's terminal.
 - If you specify Y (default), Model 204 error messages are displayed.
 - If you specify N, error messages are suppressed, but are still printed on the audit trail.

Procedure Specifications screen

Enter specifications for subsystem procedures on the Procedure Specifications screen.

```

SUBSYSMGMT          Procedure Specifications
                      Update Mode

Subsystem Name:  CARS

PROCEDURE PREFIXES
Non-compiled:   ☐
Precompiled:    ☐

PROCEDURE NAMES
Initialization:
Login:
Error:

GLOBAL VARIABLES
Command Line Variable:
Communications Variable:
Exit Value:
Error Variable:
====>

1=HELP      2=      3=QUIT    4=OPERation  5=      6=SYSClass
7=          8=          9=USERdef  10=         11=FILEuse  12=END

LU002 PLU
    
```

Figure 8-4. Procedure Specifications screen

Commands

- PF4, PF6, PF9, and PF11 each validate, save the input, and provide the next secondary screen selection.
- You must use a PF key to quit, save input, or move to a different screen.
- ENTER validates the screen.

Procedure prefixes

The *PROCEDURE PREFIXES* option requires each procedure used in a subsystem to have a prefix or be included in a procedure with a prefix.

Subsystem procedures can contain Model 204 commands, requests, continued requests, or sections of User Language code in any combination. A procedure containing commands cannot be precompiled. Procedures included in a precompiled procedure are also precompiled.

- *NON-PRECOMPILED* specifies a prefix that identifies procedures that are compiled each time they are invoked.
- *PRECOMPILED* specifies a prefix that identifies precompiled procedures, which are included in the in-core dictionary for reuse. Precompiled procedures save CPU and elapsed time.

Procedure names

PROCEDURE NAMES require assigned prefixes that indicate if the procedure is or is not precompiled:

- *INITIALIZATION* (optional) is the procedure invoked once when the subsystem is first started.
- *LOGIN* (required) is the name of the first procedure executed for every user.

Model 204 automatically includes a subsystem login procedure when a user enters a subsystem. This procedure must issue *UTABLE* commands to set compiler table sizes.

Control must be passed to an initial main menu screen if the subsystem is an Online application.

- *ERROR* (optional) is the name of the procedure invoked if an error occurs during execution of a subsystem, or if an attention interrupt occurs when no *ON ATTENTION* unit is active.

If no error procedure is supplied, Model 204 disconnects the user from the subsystem when an error occurs.

Global variables

GLOBAL VARIABLES enable you to pass information from one request to another and conditionally include procedures at the Model 204 command level. For more information, see the *Rocket Model 204 User Language Manual*.

- *COMMAND LINE VARIABLE* specifies the name of an optional global variable containing parameter input that you can enter upon logon. The command line can contain up to 255 characters.

When a user logs in to a subsystem by entering the subsystem name followed by a number of parameters, the parameter input is placed into a global variable that is available to the subsystem procedures. If not defined, the command line is discarded.

Command line variables are not destroyed when control is transferred to another subsystem through the subsystem transfer (*XFER*) facility.

- *COMMUNICATIONS VARIABLE* (required) passes control from one procedure to the next. Each procedure sets the communications variable to the name of the next procedure executed in the subsystem.

You can transfer control from one procedure to another by using a user-designated global variable.

You can transfer control between subsystems by using the reserved global variable *XFER*. For more details about transferring control, see the *Rocket Model 204 User Language Manual*.

When the application is ready to terminate, a subsystem procedure must set the communications global to the exit value.

- *EXIT VALUE* (required) specifies the setting of the communications variable for exiting the subsystem.

- **ERROR VARIABLE** (required) specifies the variable in which an error code is stored. The value of the error variable, which is used by the error procedure, indicates the type of error.

Subsystem File Use screen

The Subsystem File Use screen defines the files, including the procedure file used by the subsystem.

The screenshot shows the 'Subsystem File Use' screen with the following details:

- SUBSYSMGT** (top left)
- Subsystem File Use** (top center)
- Update Mode** (top right)
- Subsystem Name: SALES** (left)
- From:** (right)
- File/Group Name** (left column)
- File Location** (second column)
- Group Y/N** (third column)
- Auto Y/N** (fourth column)
- Mandatory Y/N** (fifth column)
- Procs NUMLK** (sixth column)
- Deferred Name** (seventh column)
- Ordered-index Deferred Name** (eighth column)

File/Group Name	File Location	Group Y/N	Auto Y/N	Mandatory Y/N	Procs NUMLK	Deferred Name	Ordered-index Deferred Name
PR SALESPRC		N	Y	Y			
1: PAYROLL		N	Y	Y			
2: EMPLOYEE		N	Y	Y			
3: CLIENTS	DALLAS	N	Y	Y			
4: PRODUCTS	DALLAS	N	Y	Y			
5: SALESGRP		Y	Y	Y			

====>

1=HELP 2= 3=QUIT 4=OPERation 5=PROCEDURE 6=
7=BACKward 8=FORward 9=USERdef 10= 11=SYSClass 12=END

LU008 PLU

Figure 8-5. Subsystem File Use screen (PQO client definition)

If you define the procedure file as a multiple procedure group, all members of the group are searched for subsystem procedures. You can control the number of group members searched, by specifying the search number on the Subsystem File Use screen NUMLK parameter. NUMLK restricts the search to the last specified number of group members.

The in-core procedure dictionary is built during initialization by searching for locked members of a procedure file group that have the precompiled prefix. If a subsystem procedure is present in more than one member of a group, only the first procedure is included in the in-core procedure dictionary.

You can modify procedures in the included members without stopping the subsystem, but you can save them in only an unlocked member of the procedure file group. You can add new procedures to any member of a procedure file group, but only procedures added to unlocked members are visible to the subsystem without restarting the subsystem.

The PROCFILE option of the CREATE GROUP command is described in the *Model 204 Parameter and Command Reference*.

Substituting a temporary group for the permanent group defined to the subsystem, defining a multiple procedure file to the \$LSTPROC function, and restrictions on using certain commands with multiple procedure files are discussed in the *Model 204 User Language Manual*.

Note: Non-system managers can add files to a subsystem if they have been assigned modify privileges (see “Administrative Privileges screen” on page 220).

Commands

- PF4, PF5, PF9, and PF11 each validate, save the input, and provide the next secondary screen selection.
- You must use a PF key to quit, save input, or move to a different screen.
- PF7 and PF8 provide scrolling to and from the first to last Subsystem File Use screen when more than one is used. Input is validated before the screen scrolls.
- ENTER validates the screen.
- You must issue the TOP, BOTTOM, and MAXIMUM commands on the command line. These commands have no corresponding PF keys.
 - TOP scrolls to the first file.
 - BOTTOM and MAXIMUM scroll to the last file.
 - MAXIMUM used with PF7 and PF8 is equivalent to TOP and BOTTOM, respectively.

File specifications

- You must specify file or group names in the column labeled File/Group Name.
 When using a group procedure file, the name specified must correspond to the permanent group. Note that an individual user can create or open a temporary group having the same name as the permanent group defined to the subsystem. If the temporary group is open before login to the subsystem, the subsystem uses the temporary group instead of the defined permanent group.
 The following restrictions apply to the use of temporary groups as application subsystem procedure files:
 - TEST or DEBUG privilege is required.
 - The last files of the permanent group must correspond exactly to the last files of the temporary group. The number of files used in both groups is specified on the NUMLK parameter.
- *FILE LOCATION* (PQO only) specifies the location of each file. If this new field is omitted, then the file is assumed to be local. Specify location only for client subsystems.
- *GROUP* (Y/N) indicates whether or not the data file is a group. The default is N.

- *AUTO* Y/N indicates automatic members (opened automatically at subsystem startup).
- *MANDATORY* Y/N indicates mandatory members (which must be open to access the subsystem).
- *PROCS* specifies the name of the procedure file or group for the subsystem. You can use up to eight characters for each entry.
- *NUMLK* specifies the number of files in a group that participate in procedure locking. Valid values are between 0 and 255, but must be less than the number of files in the group.

If Group is N, the value of NUMLK must be NULL. If Group is Y, NUMLK must be given a value.

- *DEFERRED NAME* specifies an optional deferred update data set that can be a z/OS ddname, a z/VSE DLBL name, or a CMS FILEDEF name. If a deferred name is specified, the file is opened in deferred update mode when the subsystem is started.
- *ORDERED-INDEX DEFERRED NAME* specifies an optional Ordered Index deferred update data set that can be a z/OS ddname, a z/VSE DLBL name, or a CMS FILEDEF name. If an ordered-index deferred name is specified, the file is opened in deferred update mode when the subsystem is started.

Subsystem Classes screen

Define command and file privileges for each class of subsystem users on the Subsystem Classes screen. Each class of user requires a separate screen. User class privileges defined to the subsystem override settings for OPENCTL and file privileges that reside in the password table.

Only the system manager can update the Logon Privilege, Record Security ID, and Account fields.

SUBSYSMGMT		Subsystem Classes					
		Update Mode					
Subsystem Name: SALES		From:	Class 2:		UPDATE		
Command Privileges:		Start: Y	Stop: Y	Test: N	Debug: N		
Login Privilege:		Record Security ID:			Account:		
File/Group		File Privileges					
Name	Location	Prcldef	Privdef	Sellvl	Readlvl	Updtlvl	Addlvl
PR SALESPRC		0	BFFF	0	0	0	0
1: PAYROLL		0	07E7	0	0	0	0
2: EMPLOYEE		0	07E7	0	0	0	0
3: CLIENTS	DALLAS	0	07E7	0	0	0	0
4: PRODUCTS	DALLAS	0	07E7	0	0	0	0
5: SALESGRP		0	BFFF	0	0	0	0
====>							
1=HElp		2=FILEuse		3=QUIt		4=OPeration	
7=BACKward		8=FORward		9=USErdef		10=PREVclss	
				11=NEXTclss		12=END	
LU008 PLU							

Figure 8-6. Subsystem Classes screen (PQO client definition)

Commands

- PF2, PF4, PF5, and PF9 each validate, save the input, and provide the next secondary screen selection.
- Use a PF key to quit, save input, or move to a different screen.
- PF6 displays current PRIVDEF parameter settings and lists options, described in “Security specifications” on page 212.
- PF7 and PF8 provide backward and forward scrolling when the list of files or groups exceeds one page. Input is validated before the screen scrolls.
- PF10 returns to the previous subsystem class. Input is validated and saved before moving to a different class.
- PF11 moves to the next subsystem class. Input is validated and saved before moving to a different class.
- ENTER validates the screen.
- You must issue the TOP, BOTTOM, and MAXIMUM commands on the command line. These commands have no corresponding PF keys:
 - TOP scrolls to the first file in this class.
 - BOTTOM scrolls to the last file in this class.
 - MAXIMUM used with PF7 or PF8 is equivalent to TOP or BOTTOM.

SUBSYSTEM NAME and CLASS

- *SUBSYSTEM NAME* specifies the name of the subsystem affected by the definitions entered on the screen.
- *SUBSYSTEM CLASS* is a system-assigned numeric identifier that specifies the user class being defined. As each new class is added, numeric identifiers are increased by an increment of 1.

In a semi-public subsystem, the first class is automatically assigned the class name DEFAULT, which you can change.

Security specifications

- *COMMAND PRIVILEGES* determine if the user class (SCLASS) can issue the subsystem commands START (Y/N), TEST (Y/N), STOP (Y/N), and DEBUG (Y/N). The default for all fields is N.

DEBUG specifies entering the subsystem in DEBUG mode.

- *LOGIN PRIVILEGE* (optional) specifies the user class login privileges for all, some, or none of the subsystem user classes.

Any individual user login privilege held at entry to the subsystem or specified on the Operational Parameters screen is overridden. The original value is restored when the user exits the subsystem.

The privilege option is expressed as a hexadecimal value within the range of X '00' to X 'FF', as described in "PRIVDEF parameter settings" on page 270.

NULL indicates default privileges from the Operational Parameters screen.

- *RECORD SECURITY ID* (optional) overrides any individual user Record Security ID held on entry to the subsystem. You can enter a maximum of eight characters.

NULL indicates the security activated on login.

The original value of the record security ID is restored when the user exits the subsystem.

- *ACCOUNT* (optional) specifies an account associated with specific user classes. ACCOUNT can be specified for all, some, or none of the user classes. You can enter up to 10 characters.

Any individual value of ACCOUNT held at entry to the subsystem or specified on the Operational Parameters screen is overridden. The original value is restored when the user exits the subsystem.

NULL indicates the user's login account or the value from the Operational Parameters screen.

- *FILE/GROUP* specifies files or groups that can be accessed by the defined class of users.

Define the set of files or groups belonging to the subsystem on the File Use screen and display them on this screen.

- *LOCATION* (PQO only) refers to the node where a file is located. This sample client definition shows CLIENTS and PRODUCTS at location DALLAS.
- *PRCLDEF* specifies Model 204 procedure security. Values must be between 0 (default) and 255.
 - 0 specifies no procedure security.
 - 255 specifies the highest security.
- *PRIVDEF* specifies file privileges. Values are hexadecimal 0 to X 'BFFF' (default).
- *SELLVL*, *READLVL*, *UPDTLVL*, and *ADDLVL* specify values for field-level security parameters. Values must be between 0 (default) and 255.
 - 0 specifies all users can access field values.
 - 255 restricts access of field values to users having certain privileges.
- *PROCS* specifies the name of the procedure file or group for the subsystem.

User Definitions screen

List user classes on the User Definitions (Userdef) screen for updating or browsing class definitions.

```

SUBSYSMGMT                                User Definitions
                                         UPDATE MODE

Subsystem Name: XREF

                                         Subsystem Classes

      1 USERS
      2 ADMIN

ENTER SUBSYSTEM CLASS NUMBER: █

====>
SUM009 The default subsystem class is displayed as bright.
1=HELP    2=FILEuse  3=QUIT    4=OPERation  5=PROCEDURE  6=SYSCLASS
7=BACKward 8=FORward  9=       10=       11=       12=END

LU002 PLU
  
```

Figure 8-7. User Definitions screen

Commands

- PF2, PF4, PF5, and PF6 display the next secondary screen.

- PF7 and PF8 allow scrolling backward and forward for listing next or previous subsystem classes.
- ENTER displays the Subsystem Class Users screen for the class selected.
- You must issue the TOP, BOTTOM, and MAXIMUM commands on the command line. These commands have no corresponding PF keys:
 - TOP scrolls to the first class.
 - BOTTOM scrolls to the last class.
 - MAXIMUM used with PF7 and PF8, is equivalent to TOP and BOTTOM, respectively.

Screen specifications

- *SUBSYSTEM NAME* specifies the applicable subsystem.
- *SUBSYSTEM CLASSES* displays a listing, by class number and name, of the user classes defined on the Subsystem Classes screens.
- *SUBSYSTEM CLASS NUMBER* specifies the Subsystem Class number to be updated.

Subsystem Class Users screen

Define user accounts assigned to a specific class on the Subsystem Class Users screen. A user can belong to only one class within a subsystem. If using more than one subsystem, the user can belong to a different class (different privileges) in each subsystem.

```

SUBSYSMGMT                      Subsystem Class Users
                                UPDATE MODE

Copy from      Sclass:          Subsystem:        From:
               Sclass:  UPDATE   Subsystem:  SALES    From:

       Account slots: 1 - 48     Number of accounts:    4

> ADMIN01           >                               >
> SABROWN          >                               >
> SAKURTH           >                               >
> SASAH            >                               >
>                  >                               >
>                  >                               >
>                  >                               >
>                  >                               >
>                  >                               >
>                  >                               >
>                  >                               >
>                  >                               >
>                  >                               >
>                  >                               >
====>

1=HELP         2=              3=QUIT      4=DELusers    5=PREview      6=
7=BACKward    8=FORDward     9=             10=COPYusers   11=CPReplace   12=END

LU008 PLU

```

Figure 8-8. Subsystem Class Users screen

Duplicate account entries result in the display of the class in which the original name resides.

Users can be added to a class, deleted, or replaced by entering, deleting, or typing over the individual Model 204 user account identification after the prompt (>).

COPY SCLASS USERID option

The COPY SCLASS USERID options allow you to copy user IDs from an SCLASS already defined in one subsystem to the current SCLASS being defined.

PF keys

The COPY SCLASS options are represented by four PF key functions on the Subsystem Class Users screen:

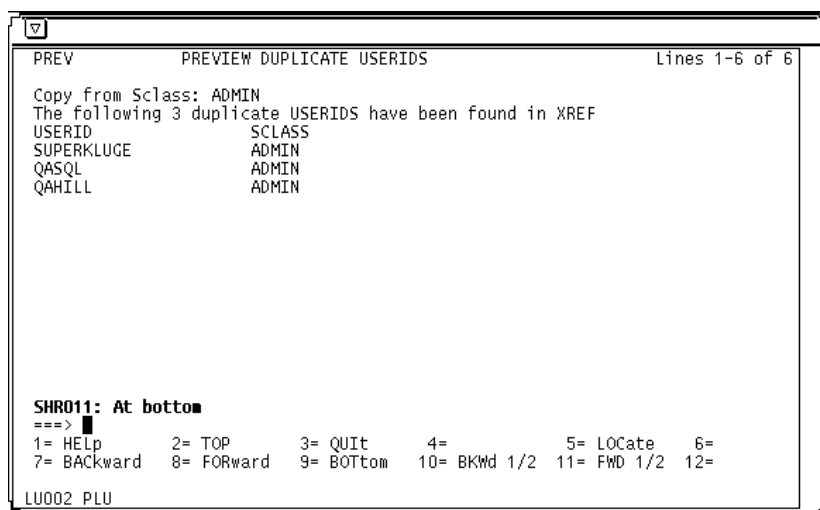
- *PF4 (DELusers)* deletes all users from a particular SCLASS selected from the User Definitions Screen. To execute this command, press PF4. A warning message appears with the name of the SCLASS from which users are about to be deleted. Reenter the command if you are sure that you want to perform the Delete.
- *PF5 (PREview)* displays a list of all duplicate user IDs common to the copy-from SCLASS and any of the SCLASSES in the copy-to subsystem. The list is displayed on a separate screen, in the section “Sample Preview screen” on page 215.
- *PF10 (COPyusers)* copies user IDs from the SCLASS specified in the copy-from field to the SCLASS selected from the User Definitions screen. This SCLASS is displayed on the current screen as a protected field.

When duplicate user IDs exist in another SCLASS in the copy-to subsystem, a warning message indicates that duplicates have not been copied. As with PF4, reenter the command to perform the copy.
- *PF11 (CPReplace)* copies user IDs as COPyusers does, but replaces duplicate user IDs. When duplicates exist, a warning message indicates that duplicates are moved from existing SCLASSES. Reenter the command to perform the copy with replacement.

- Other PF keys function as in other secondary screens.

Sample Preview screen

The following screen is an example of the screen that appears when you execute the PREview command (PF5) on the Subsystem Class Users screen.



```
PREV          PREVIEW DUPLICATE USERIDS          Lines 1-6 of 6

Copy from Sclass: ADMIN
The following 3 duplicate USERIDS have been found in XREF
USERID          SCLASS
SUPERKLUGE      ADMIN
QASQL           ADMIN
QAHILL          ADMIN

SHR011: At bottom
===>
1= HELP      2= TOP      3= QUIT      4=          5= LOCate    6=
7= BACKward  8= FORWARD  9= BOTTOM    10= BKWd 1/2 11= FWD 1/2 12=

LU002 PLU
```

Figure 8-9. Preview Duplicate USERIDS screen

This screen is purely informational. Press PF3 to return to the Subsystem Class Users screen.

Subsystem Import/Export options

The Import and Export functions allow an authorized user to migrate subsystem definitions from one Model 204 environment to another via the migration file D204SYS, a Model 204 file created during Dictionary installation.

The Export Delete function deletes a specified subsystem from the D204SYS file. EXPortlist displays the list of subsystems that currently reside in D204SYS for which the user has ALL privileges (otherwise the user is not authorized to export them).

The four Import/Export options are all accessible from the Activity screen, shown below. Import, Export, and Export Delete are options 7, 8, and 9, respectively. You can display an Exportlist by pressing PF11.

```

SUBSYSMGMT      Subsystem Management Facility      VER 3 REL 2.0I
                  1 Select Subsystem Activity
                  -----
                  1. Add
                  2. Modify
                  3. Browse
                  4. Copy
                  5. Rename
                  6. Delete
                  7. Import
                  8. Export
                  9. Export Delete
                  10. ADMIN

Subsystem Name: SALES  From:
Copy/Rename To:      From:
Export Users ? : N

====>

1=HELP      2=FILEuse  3=QUIT   4=OPERation  5=PROCEDURE  6=SYSclass
7=          8=        9=USERdef  10=         11=EXPORTlist 12=

LU008 PLU

```

Figure 8-10. Activity screen

Export

The Export option allows authorized users to export subsystem definitions to another Model 204 Dictionary. To export a subsystem definition:

1. Select Option 8 from the Activity screen.
2. Specify the subsystem to be exported.
3. If you want to export SCLASS users, change the default value N to Y on the Export Users line.
4. Press Enter.

A message appears at the bottom of the screen indicating whether the Export has been successful. If the subsystem has already been exported by another user, then you cannot export it until it is deleted from D204SYS (see “Export Delete”).

Import

To import subsystem definitions, you must enter SUBSYSMGMT in the Dictionary to which the definition is to be imported. If the subsystem already exists, you must delete or rename it before importing. To import a subsystem definition:

1. Select Option 7 from the primary menu.
2. Specify the subsystem to be imported

3. Press Enter.

A message appears at the bottom of the screen indicating whether the Import was successful. After an Import, the subsystem definitions remain in D204SYS until they are deleted.

Export Delete

To delete a subsystem from D204SYS:

1. Select Option 9 from the primary menu
2. Specify the subsystem to be deleted from D204SYS
3. Press Enter.

Exportlist

To display the subsystems in D204SYS for which you have ALL privileges, press PF11. The list displayed shows the name of each subsystem, the date and time of export, and the exporting user ID.

Exportable data: scope and limitations

The Import/Export functions export all information in METADATA and DATALINK except for STAGED entities and user-defined links to the subsystem.

After an import, the LAST UPDATED field for a METADATA entry is changed to the date of import, and the UPDATED-BY field is changed to the importer's user ID.

User SCLASSES can be imported (see "Export" on page 217); but administrative privileges cannot be imported.

Subsystem Administration screen

The Subsystem Administration screen appears when the system manager selects option 10 on the SUBSYSMGMT primary menu. This option is available only to a system manager.

```

SUBSYSMGMT          Subsystem Administration

  [ ] Usage (U) OR Admin (A) privileges
-----
      _ Enter selection number

      1. Define Privileges
      2. Copy
      3. Rename
      4. Delete

User Account:
Copy/Rename To:
Subsystem pattern:

===>

1=HElp      2=      3=QUIT  4=      5=      6=
7=          8=          9=      10=     11=     12=

LU002 PLU

```

Figure 8-11. Subsystem Administration screen

Select Usage Privileges by entering **U** on the space provided (**U** is the default). These privileges correspond exactly to the User Activity privileges in previous releases of Model 204. Selecting **U** brings up the User Matrix screen described in the section “User Matrix screen” on page 223

Select Administrative Privileges for a User Account by entering **A**. The four Admin options are explained in the following section, “Admin options”.

When setting privileges for specific users, remember that access to SUBSYSMGMT must be authorized via the DICTADMIN facility, as it is for every Model 204 Dictionary subsystem. The privileges granted here supplement but do not override general access privileges granted through DICTADMIN.

Admin options

The Subsystem Administration screen offers four options, described in the following sections.

1. Define Privileges

The Define Privileges option lets the system manager grant administrative privileges to specific users for specific subsystems. Selecting this option brings up the Administrative Privileges screen, shown in Figure 8-12 on page 221. To execute Define Privileges, enter 1 at the selection number prompt, specify the user ID to be assigned privileges in the User Account field, and press Enter.

2. Copy

The Copy option duplicates a user's administrative privileges to another user ID. To execute Copy, enter 2 at the selection number prompt, specify the user ID to be copied in the User Account field, specify the name of the user ID to be copied to in the Copy/Rename field, and press Enter.

3. Rename

The Rename option changes a user ID associated with a given set of administrative privileges. To execute Rename, enter 3 at the selection number prompt, specify the user ID to be renamed in the User Account field, specify the new name in the Copy/Rename field, and press Enter.

4. Delete

The Delete option takes away all administrative privileges associated with a specific user ID. To execute Delete, enter 4 at the selection number prompt, specify the user ID in the User Account field, and press Enter.

Subsystem Pattern field

The Subsystem Pattern field applies to all options in both Usage and Admin modes. You can leave it blank, enter a single subsystem name, or enter a pattern.

- If you leave this field blank, the next screen displays all subsystems.
- If you select a pattern following pattern specifications described in the *Model 204 User Language Manual*, then the found set of subsystems are displayed on the next screen—the User Matrix screen, if in Usage mode; the Administrative Privileges screen, if in Admin mode.

Administrative Privileges screen

The Define Privileges option selected in Admin mode displays the Administrative Privileges screen. To create or delete privileges, type or space over Xs in the appropriate spaces.

Note: Take care when granting All and Modify privileges. A user with these privileges can update subsystem file use data.

SUBSYSTEM **Administrative Privileges**

Account: DVDAN (Select privileges with an X)

Subsystem	All	Modify	Browse	Udef
ALL SUBSYSTEMS	-	-	-	-
CARS	-	-	-	-
RPI	-	-	-	-
XREF	-	-	-	-

====> █

1=HElp 2= 3=QUIT 4= 5=LOCate 6=
7= 8= 9= 10= 11= 12=END

LU002 PLU

Figure 8-12. Administrative Privileges screen

Hierarchy of privileges

The ALL SUBSYSTEMS line on this screen grants the account the selected privileges for all subsystems, including: All, Modify, Browse, Udef.

If both ALL SUBSYSTEM and specific subsystem privileges are granted, the higher privilege takes precedence.

For Import, Add, and Copy privileges, a user must also be a member of the ADDPRIV SCLASS, described in the section “ADDPRIV SCLASS and Add Privileges” on page 222.

PF keys

- *PF1 (HElp)* displays help text for the screen.
- *PF2 (TOP)* scrolls to the top of the display.
- *PF3 (QUIT)* returns the user to the Admin Screen without saving changes. Before exiting, warning messages are displayed indicating that changes have not been saved.
- *PF5 (LOCate)* locates a specified subsystem on the display list. Specify the name on the command line before pressing PF5. If the subsystem name is found, it is highlighted and displayed at the top of the list. If it is not found, the display is unchanged.
- *PF7 (BACKward)* scrolls backward through the list of subsystems specified. A maximum of ten subsystems are displayed on the screen at once.
- *PF8 (FORward)* scrolls forward through the list of subsystems specified.

- *PF12 (END)* saves all screen changes and returns the user to the Admin Screen.
- The Enter key has no effect on this screen.

ADDPRIV SCLASS and Add Privileges

To give a user Add, Copy, or Import privileges, the system manager must add the appropriate user ID to a new SCLASS called ADDPRIV. Do this as shown in Figure 8-13 on page 223, the User Matrix screen, or in Figure 8-7 on page 213, the User Definitions screen.

Summary of subsystem privileges

Table 8-6 shows how the system of assigned privileges relates to the ten functions displayed Figure 8-1 on page 198, the Subsystem Management Facility screen. The columns are privileges granted, including general system manager privileges. The rows are SUBSYSMTGMT functions.

Table 8-6. Summary of subsystem privileges

Option	Assigned privilege					
	ALL	Modify	Browse	Udef	ADDPRIV member	System manager
1. Add					X	X
2. Modify	X	X		X		X
3. Browse	X	X	X			X
4. Copy	X				X	X
5. Rename	X					X
6. Delete	X					X
7. Import					X	X
8. Export	X					X
9. Export Delete	X					X
10.Admin						X

Note: Copy privileges require both ALL and ADDPRIV SCLASS membership. A user assigned Udef privileges can update only the Userdef screens.

User Matrix screen

To change single account in any class of a subsystem, use the User Matrix screen. Access the User Matrix screen by selecting the ADMIN option (number 10) on the Activity screen, then selection 1 under the Usage options.

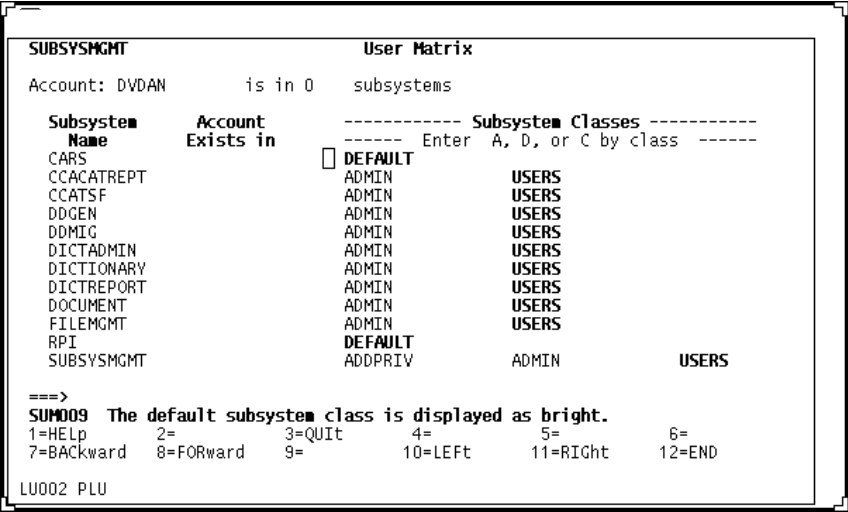


Figure 8-13. User Matrix screen

- Commands**
- PF7 and PF8 provide scrolling to and from the first to last Subsystem Class Users screen when more than one screen is required.
 - PF10 and PF11 provide scrolling to the left and to the right to locate all subsystem classes for the subsystem shown in the leftmost column.
- Specifications**
- SUBSYSTEM NAME* specifies the subsystem in which the user account is defined.
 - ACCOUNT EXISTS IN* specifies the class in which the account is defined.
 - SUBSYSTEM CLASSES* commands are entered in a one-character input field that precedes each class displayed:
 - A** specifies adding the current account to the class in the subsystem shown.
 - C** specifies changing the class in the subsystem to which the current account belongs from the ACCOUNT EXISTS IN or default class to the class at the right of the command.
 - D** specifies deleting the current account from the class in the subsystem shown. Note that this is one way to change an account's class from a nondefault to the default class in a semi-public subsystem.

Dynamic APSY support

An application subsystem (APSY) is comprised of Model 204 procedures executed in a logical order. You define the *operation* of the application using the Model 204 Subsystem Management facility (SUBSYSMTGMT). The Model 204 procedures determine the *contents* of the application. You can dynamically modify the following aspects of an APSY, without terminating the APSY and without interrupting service to your users.

- A subset of application subsystem attributes.
- Precompiled and non-precompiled procedures in files that participate in procedure locking while the subsystem is active.
- Save compiled procedures that contain commands by saving the commands before and after the BEGIN/END. The commands are saved on a chain of CCATEMP pages in the form of a temporary procedure. The temporary procedure contains pointers to these pages. These temporary procedures are not the user-accessible temporary procedures (0, -1, and so on) and do not interfere with their operation.

Introducing dynamic APSY subsystem attribute changes

Subsystem attribute modifications are saved to the permanent subsystem definition in the CCASYS file as well as to the active in-core subsystem definition without interrupting service to the subsystem user.

Updating an active subsystem via SUBSYSMTGMT

The SUBSYSMTGMT subsystem maintains all subsystem definitions. You can modify various attributes of a subsystem through a variety of screens provided by SUBSYSMTGMT. Once you are satisfied with the modifications that are made, press <PF12> to end the updates and to save the definition to disk.

To support modifications to the active in-core definition, the SUBSYSMTGMT subsystem was changed as follows. When you press <PF12> to save the updates, SUBSYSMTGMT checks to see if the subsystem is currently active.

- If the subsystem is not active, you return to Subsystem Management Facility screen.

- If the subsystem is active, the following Active Subsystem Update screen is displayed:

```

SUBSYSTEMGMT                Active Subsystem Update

                               Subsystem: MCC1

                               is currently active.

                               Would you like to update the active subsystem definition?

                               PF3 to QUIT/Not update the active subsystem definition

                               PF12 to END/Update the active subsystem definition

=====>

1=HElP      2=          3=QUIt   4=          5=          6=
7=          8=          9=          10=         11=         12=END

```

At this point, the permanent definition in the CCASYS file has been updated. Your response to this screen determines whether the subsystem will be *dynamically* updated.

- To update the active subsystem definition, press <PF12>.
- To keep the active in-core definition unchanged, press <PF3>.

Only the <PF1> (HElP), <PF3> (QUIt), and <PF12> (END) function keys are active at this time.

Using Active Subsystem Help

If you press <PF1> while in the Active Subsystem Update screen, the following Active Subsystem Help screen is displayed:

```
SUBSYSMGMT          Active Subsystem Help

Use this screen to update the active subsystem definition. If the subsystem
has been started and is currently active then the user has the option of
refreshing the subsystem definition that is currently running. Only the
following fields are supported for active update at this time:

. Log user into M204
. Log user out of M204
. Auto Commit
. Maximum Iterations
. Account
. Disconnect
. Informational
. Error
. Login Procedure
. Error Procedure

In order to update APSY fields that are not supported for active update,
the user must stop and restart the subsystem.
Use the PF key below to move to the next screen:
PF 3 = QUIT          Does NOT UPDATE active subsystem definition.
                     Returns to Subsystem Management primary screen.
PF12 = END           Updates active subsystem definition.
                     Returns to Subsystem Management primary screen.
```

Leaving the active subsystem unchanged

If you press <PF3> or enter QUIT in the Active Subsystem Update screen, you leave the active subsystem unchanged and return to the Subsystem Management Facility screen. The changes you made go into effect the next time the subsystem is started

Updating the active subsystem

Press <PF12> or enter END to update the active subsystem definition. The changes to the dynamic attributes go into effect immediately. Non-dynamic changes do not take effect until the next time the subsystem is started.

- If the update is successful, you are returned to SUBSYSMGMT main menu screen without any confirmation messages displayed.
- If the update is unsuccessful, one of the following messages is displayed on the main menu screen.

```
SUM014 Unable to update active definition for sub-
system: name
```

```
SUM015 Unable to update active definition, name is no
longer active
```

Understanding the Dictionary/204 data definition errors

SUM014 is displayed when an error occurs while attempting to update the active definition. The reason for the error can be found on the audit trail with one of the following errors:

M204.1457 UNABLE TO SCAN LIST OF SUBSYSTEM names

M204.1685 SUBSYSTEM name DOES NOT EXIST

M204.2253 SUBSYSTEM name, record type - RECORD CONTAINS INVALID DATA

M204.2391 SUBSYSTEM name, record type - TRANSLATION FAILED FOR FIELD fieldname

M204.2395 SUBSYSTEM name, record type - RECORD MISSING

SUM015 is displayed if the subsystem is stopped while attempting to update the active definition. The following error message is generated on the audit trail:

M204.2647 UNABLE TO UPDATE ACTIVE DEFINITION, <name> IS NO LONGER ACTIVE.

Limits to dynamic subsystem attribute changes

The Table 8-7 lists the subsystem attributes that you can change. You can make changes on the Operational Parameters screen in SUBSYSMGMT:

Table 8-7. Operational Parameters you can change

Operational Parameter	Specifies
Log user into M204	The user is logged into Model 204 with the subsystem name as the USERID
Log user out of M204	The user is logged out upon leaving subsystem
Auto Commit	A User Language COMMIT/RELEASE statement is run at the end of each procedure
Maximum Iterations	Maximum number of consecutive times you can invoke the same procedure before the ERROR procedure is invoked
Account	The account value other than the one used at logon
Disconnect	Whether to display a system disconnect message
Informational	Whether to display Model 204 informational messages
Error	Whether to display Model 204 error messages

You can change the following on the Procedure Specifications screen in SUBSYSMGMT:

Procedure	Description
Login	First procedure that is invoked for every user
Error	Procedure that is invoked if an error occurs when a subsystem is running

If you dynamically change the name of the Login procedure or the Error procedure, the new name must be defined in the in-core procedure dictionary. You cannot add new procedures to a subsystem procedure file that participates in procedure locking once the subsystem has been started and have those procedures included by the subsystem. Therefore, if you change the name of the login or error procedure that resides in a file that participated in procedure locking, then that procedure must have existed when the subsystem was started.

Dynamically refreshing procedure compilation with the REFRESH SUBSYSPROC command

The REFRESH SUBSYSPROC command discards the existing precompilation and precompiles the first time through for each SYSCCLASS. Once a procedure is refreshed, the subsystem includes the new version of the procedure the next time that procedure is invoked.

The newer version of the procedure is compiled and saved in CCATEMP for subsequent execution. For the correct syntax, see the *Rocket Model 204 Parameter and Command Reference*, "REFRESH SUBSYSPROC: Replace procedures in active subsystem".

The REFRESH SUBSYSPROC command optionally copies a procedure from an open file (or group) into another open subsystem file (or group) and updates the APSY in-core procedure dictionary with the new procedure text page. In group context, the REFRESH SUBSYSPROC command replaces and refreshes only procedures that participate in procedure locking, based on the NUMLK parameter.

Generating success messages

Successful execution of the REFRESH SUBSYSPROC command generates the following confirmation messages:

```
M204.2665 procname REFRESHED IN SUBSYSTEM subsystem-name
```

If a FROM clause is specified, the following message is produced:

```
M204.2666 procname REPLACED IN FILE filename [IN GROUP
groupname]
```

Generating error messages

The REFRESH SUBSYSPROC command may fail under the following circumstances:

- When you do not have Refresh privileges to issue the command, the command fails, the procedure is not refreshed, and the following message is generated:

```
M204.0930 REQUIRES SUBSYSTEM COMMAND PRIVILEGE
```

- When the procedure is not defined in an active subsystem, the command fails, the procedure is not refreshed, and the following message is generated:

```
M204.2668: procname NOT FOUND IN ANY ACTIVE SUBSYSTEM
```

- If a procedure is not found in the file named in the FROM clause, or if the user does not have appropriate file privileges to copy procedures, the following message is generated:

```
M204.1158: CAN'T COPY PROCEDURE: procname
```

- When the subsystem file (or group) is out of space to copy the procedure, the command fails, the procedure is not refreshed, and the following message is generated:

```
M204.1483: NOT ENOUGH TABLED SPACE TO STORE PROCEDURE
```

- If another user is processing the procedure, the following message is generated:

```
M204.2669: PROCEDURE procname IS IN USE BY SUBSYSTEM  
subsys-name
```

Understanding REFRESH SUBSYSTEM command privileges

Privileges to issue the REFRESH SUBSYSTEM command are set in the SUBSYSMGMT subsystem. A new field, named REFRESH, is defined on the Subsystem Classes screen on the Command Privileges line. The input to this field is either Y or N, where N is the default. To issue the REFRESH SUBSYSTEM command, set the value of this field to Y.

The following screen illustrates the Refresh privilege set to Y:

SUBSYSMTGMT		Subsystem Classes					
		Update Mode					
Subsystem Name: MCC		From:	Class	1:	DEFAULT	DEFAULT	
Command Privileges.		Start: Y	Stop: Y	Test: Y	Debug: Y		
		Resume: Y	Suspend: Y	Refresh: Y			
Login Privilege:		Record Security ID:		Account:			
File/Group		----- File Privileges -----					
Name	Location	Prcldef	Privdef	Sellvl	Readlvl	Updtlvl	Addlvl
PR	PROCS	0	BFFF	0	0	0	0
1:	DATA	0	BFFF	0	0	0	0
2:							
3:							
4:							
====>							
1=HELp	2=FILEuse	3=QUIt	4=OPERation	5=PROcedure	6=PRIVdef		
7=BACKward	8=FORward	9=USERdef	10=PREVclass	11=NEXTclass	12=END		

Subsystem processing

When you refresh an active subsystem procedure, the following occurs:

- Precompiled procedures are recompiled for each SCLASS.
- While a procedure is being refreshed, the procedure is locked. While a procedure is locked, other users cannot access it. The length of time that the procedure is stopped is as short as possible.

However, if a subsystem attempts to include a procedure that is in the process of being refreshed, the APSY tries to invoke the procedure a few times before giving up. If the APSY fails to invoke the procedure, then the subsystem error procedure is invoked with the error global set to RFR.

You must write an error procedure instructing your subsystem how to respond to various errors.

Handling a blocked refresh for a subsystem procedure

If an APSY subsystem procedure with the subsystem precompile prefix is compiled, and the procedure is found in an unlocked subsystem procedure file, the following message is generated:

```
M204.0468: COMPILATION NOT SAVED - INCLUDE FROM UNLOCKED
FILE
```


Using the SUSPEND SUBSYSTEM command

Rocket Software recommends that you issue the SUSPEND SUBSYSTEM command and wait for the active users to exit. If you do not wait, you could have the following problem. The subsystem, which has logically related procedures, is suspended and a user invokes a procedure whose higher-level or lower-level procedure has not been updated and refreshed yet. For syntax details see the *Rocket Model 204 Parameter and Command Reference*, "SUSPEND SUBSYSTEM".

Privileges to issue the SUSPEND SUBSYSTEM command are set in the SUBSYSMGMT subsystem.

If you issue a SUSPEND SUBSYSTEM command when...	The subsystem is set to...
No active users are in the application subsystem	Suspended state.
Active users are in the subsystem	Suspending state. Current, active users continue to use the subsystem, but new users cannot enter it. After all users exit the subsystem, it is set to the Suspended state.

Monitoring the subsystem

You can issue a MONITOR SUBSYSTEM command to check the status—such as Drain or Suspend—of the subsystem as well as the number of users still running. Once a subsystem is fully suspended and the number of users is zero (0), you can safely refresh a set of logical procedures.

If you suspend a subsystem that still has active users, then the STATUS option from the MONITOR SUBSYSTEM command is set to SUSPENDING. When a subsystem is fully suspended with no active users, the STATUS option from the MONITOR SUBSYSTEM command is set to SUSPENDED.

If the subsystem is in the process of suspending, the following message is generated:

```
M204.2659 subsys-name SET TO SUSPEND, REMAINING USERS=n
```

SUSPEND SUBSYSTEM command messages

The SUSPEND SUBSYSTEM command may produce the following messages:

- When a subsystem is successfully suspended, the following message is generated:

```
M204.2661 SUBSYSTEM subsys-name SUSPENDED
```

- When there are still active users in the subsystem, the following message is generated:

```
M204.2659 subsys-name SET TO SUSPEND, REMAINING USERS =  
<n>
```

- If you issue the SUSPEND SUBSYSTEM command without Subsystem Suspend privileges defined for your SCLASS, the following message is generated:

```
M204.0930 REQUIRES SUBSYSTEM COMMAND PRIVILEGE
```

- The SUSPEND SUBSYSTEM command is valid against only an active subsystem. If the subsystem has not been started, the following message is generated:

```
M204.1126 SUBSYSTEM name MUST BE STARTED
```

- If the subsystem is in TEST mode, it is locked from other users, so the following message is generated:

```
M204.0448 SUBSYSTEM TEST IN PROGRESS, COMMAND REJECTED
```

- If a subsystem is in the process of starting, but not yet fully active, the following message is generated:

```
M204.2311 SUBSYSTEM subsys-name IS BEING STARTED
```

- If the subsystem was set to Stop and is waiting for all active users to quit, the following message is generated:

```
M204.0446 SUBSYSTEM subsys-name IS TEMPORARILY DISABLED
```

- If the subsystem requires users to log into Model 204 and you are not logged in, the following message is generated:

```
M204.1031 PLEASE LOGIN
```

- If an error prevents the subsystem from being suspended, the following message is generated. The preceding message states the cause of the error.

```
M204.2656 UNABLE TO SUSPEND SUBSYSTEM subsys-name
```

Introducing the RESUME SUBSYSTEM command

When you have completed refreshing a suspended subsystem, you can reactivate it with a RESUME SUBSYSTEM command. For a brief syntax discussion, see the RESUME SUBSYSTEM command in the *Rocket Model 204 Parameter and Command Reference*.

Privileges to issue the RESUME SUBSYSTEM command are set in the SUBSYSMGMT subsystem.

The RESUME SUBSYSTEM command may generate the following messages.

- When a subsystem successfully resumes, the following message is generated:

M204.2657 SUBSYSTEM subsys-name RESUMED

- If you issue the RESUME SUBSYSTEM command without Subsystem Resume privileges defined for your SCLASS, the following message is generated

M204.0930 REQUIRES SUBSYSTEM COMMAND PRIVILEGE

- The RESUME SUBSYSTEM command is valid against only a suspended subsystem. If the subsystem was not suspended, the following message is generated:

M204.2658 SUBSYSTEM subsys-name NOT IN SUSPEND STATE

- The RESUME SUBSYSTEM command is valid against only a suspended subsystem. If the subsystem has not been started, the following message is generated:

M204.1126 SUBSYSTEM subsys-name MUST BE STARTED

- If the subsystem is in Test mode, it is inactive for other users, so the following message is generated:

M204.0448 SUBSYSTEM TEST IN PROGRESS, COMMAND REJECTED

- If a subsystem is in the process of starting, but not yet fully active, the following message is generated:

M204.2311 SUBSYSTEM subsys-name IS BEING STARTED

- If the subsystem was set to stop and is waiting for active users to quit, the following message is generated:

M204.0446 SUBSYSTEM subsys-name TEMPORARILY DISABLED

- If the subsystem requires users to be logged in to Model 204 and you are not logged in, the following message is generated:

M204.1031 PLEASE LOGIN

- If an error prevents the subsystem from resuming processing, the following message is generated. The previous message states the cause of the error.

M204.2655 UNABLE TO RESUME SUBSYSTEM subsys-name

Using precompilable procedures with commands

If you wish to conditionally compile and save a User Language request through the use of dummy string comments, then you must ensure that the value of the dummy string is the same for all User Language statements for that request. Otherwise, unpredictable results occur.

In addition, the loading user will invoke the request that was conditionally compiled by the compiling user. This behavior is simply noted as a reminder.

If a precompiled procedure issues the INCLUDE command to compile and run a User Language request the INCLUDE command is saved, not the compilation of the request that was included.

Dummy string substitution does not take place when saving commands that contain dummy strings. Instead, when the saved commands are loaded and executed, the current value of the dummy string is used. For example, if you include the following command in a precompiled procedure, whatever is currently in the global COMMAND is executed.

```
?&COMMAND
```

Procedures that include multiple BEGIN/END blocks are not eligible for precompilation.

Handling subsystem error procedures

If a subsystem error procedure is cancelled due to attempted terminal I/O and the return code is one of the following—CAN, HNG, HRD, or SFT—then the error procedure cannot attempt to issue any of the following User Language statements:

- PRINT
- READ
- READ MENU
- SCREEN
- SKIP
- \$PROMPT
- \$READ
- Or any other statement that writes to the user's terminal

All other User Language statements are permitted.

9

Customizing Functions and Translation Tables

In this chapter

- Overview
- Adding functions to the FUNU module
- Coding a function
- Encoding/Decoding facility
- Converting user-written functions

Overview

User-written User Language functions are stored in the user function module FUNU. The FUNU module is distributed as a skeletal module containing a table for functions and arguments, and the assembler code for one function. You can list the contents of FUNU using the procedures given in the *Model 204 Installation Guide* for your operating system.

The Encoding/Decoding facility uses the translation tables contained in the distributed CDTB module. The \$CODE/\$DECODE functions access the translation tables to allow the use of both coded and string values of fields. The contents of the translation tables can be listed by using the procedures listed in the *Model 204 Installation Guide* for your operating system.

You can store messages and return codes for user-written functions in the MSGU module.

This chapter gives the procedures for adding user-written functions to the FUNU module, customizing the translation tables contained in the CDTB module, and adding messages to MSGU.

If you are migrating to 31-bit or multiprocessing environments, review the conversion considerations summarized the section “Converting user-written functions” on page 243.

For general directions on assembly and link-editing, see the Rocket Model 204 Installation Guides.

Adding functions to the FUNU module

You can add user-written functions to the FUNU module before FUNU is assembled and linked into Model 204. Reassembly and relinking is required whenever you migrate to another release of Model 204.

To add functions to the FUNU module:

1. Label the function table in the FUNU source code FUNUTAB.
2. End the function table in the FUNU source code with DC X'FFFF'.

The location of the function table in the code is not significant.

3. Make the new function available to the Model 204 User Language compiler by adding an entry to the function table using the FN macro:

```
FN function name,R-type,(A-type ] , A-type ] BDB)
```

Where

- *function name* specifies a unique 1- to 7-character name of the function without the leading \$. Spaces or special characters are not allowed.

User-written functions must not duplicate the name of Rocket Software functions. A list of Rocket Software functions is given in the *Model 204 User Language Manual*.

- *R-type* represents the type of value returned from the function. Only one value can be returned. *S* represents a string value. *N* represents a numeric value.
- *A-type* describes the type of argument value. More than one argument can be specified. *S* represents a string value. *N* represents a numeric value.

In the following example, \$SEP receives three string arguments and returns a string value:

```
FN SEP,S,(S,S,S)
```

4. Code the function (as described in “Coding a function” on page 237).

Coding a function

Coding conventions

The following conventions are required for coding a user-written function:

- Begin the function code with:

```
$name ENTER
```

- End the function code with:

```
POOL
```

- Observe the following register conventions (correspondence between Rocket Software mnemonic names and IBM general registers is printed at the beginning of the module in the expansion of the STARTS macro):
 - Registers RC, R6, PD, and RL must not be modified.
 - Registers T1 through T5 and R1 through R5 can be used as work registers.
 - Registers R1 through R5 are somewhat more permanent.
- Use BUF1 and BUF2 (260 bytes each) as buffers (doubleword aligned).
- Use string, numeric, or omitted arguments.

String arguments

If the *n*th argument is type S, you can move the argument into a work area by using the following code. Registers R1 through R4 are used by this sequence:

```
ARG n
L R2,4(X1)
CCALL RTSTRL
EX R2,FOO
STC R2,BUF1
.
.
FOO MVC BUF1(0),0(R1)
.
```

- The ARG macro returns the VTBL pointer to the *n*th argument in X1.
- The RTSTRL routine sets R2 to the length of the string and sets R1 to address-1 of the string.
- The length of the string is given by register R2.
- The string is moved to BUF1, starting at location BUF1 + 1.
- The string length is moved to location BUF1.

Numeric arguments

If the n th argument of a function is type N, you can obtain the argument in floating-point form or binary integer form (truncated or rounded) using registers R1 through R3, as follows:

```
ARG n
LA R3,x
CCALL INTEG
.
```

where:

- $x = 0$, if a truncated value is required
- $x = \text{value}$, if a rounded value is required

After this sequence, F0 contains the original floating-point value. If the number is valid, R1 contains the integer portion of the number as a binary value and R2 is zero. If the number is invalid, R1 is zero and R2 is nonzero.

Omitted arguments

When writing a call to a function, you can omit an argument, as follows:

```
$READ ( )
$SUBSTR(NAME, , 5)
```

Where

- If the omitted argument is type S, the value of the argument defaults to a null string.
- If the argument is type N, the value defaults to zero.

NOARG option

You can specify the NOARG option on the ARG macro in the event that an argument is omitted. When NOARG is used, the code branches to the location indicated by the NOARG option. For example:

```
ARG n,NOARG=ALLDONE
L R2,4(X1)
CCALL RTST
.
.
.
CCALL DKRL
ALLDONE DS OH
.
```



```
.  
.   
LEAVESTR
```

You can also specify the following, which returns control to the next sequential instruction after the ARG statement:

```
NOARG = *
```

Using the ENTER macro to allocate working storage

You can use the VARS argument of the ENTER macro to define stack variables for working storage unique to each user-written function. This method of allocating storage is preferable to STORAGE, GETMAIN, and FREEMAIN instructions. Using the ENTER macro avoids the possibility of causing Online waits or having all users share the same storage.

Syntax for ENTER VARS

```
ENTER VARS = ( , X  
    (varname1,varlength1) , comments X  
    (varname2,varlength2) , comments X  
    (varname3,varlength3) , comments X  
    .  
    .  
    .  
) , AMODE=n
```

You must initialize each of these variables.

Routines available for user written \$functions

There are CCALL entry points for: DATE, DATE3, and DATE4. All routines must be called with T1 pointing to a 26-byte answer area. Rocket Software recommends that you allocate the answer area using the VARS=(name, len) pushdown list variable of the ENTER macro. Table 9-1 lists how the current date and time is stored in the area with this format:

Table 9-1. Formats used to store the date routines

Code entry point	Format used for storing...
DATE	'YY.DDD MON DD HH.MM.SS'
DATE3	'CYY.DDD MON DD HH.MM.SS'
DATE4	'YYYY.DDD MON DD HH.MM.SS'

All registers are returned intact with the exception of the DATE call. The DATE call changes only the T4 register, returning the number representing the current month (1-12).

Return values

Functions can return a string or a numeric value. The following coding considerations apply:

- To return a string value, end the function routine by doing the following:
 - Load R1 with a pointer to the location of the result string minus one.
 - Load R2 with the length of the result string.
 - Issue the LEAVESTR macro at the end of the code, just before LTORG to drop local addressability.
- To return a numeric value, end the function routine in one of the following ways, depending on how the result is computed:
 - For an integer result, end the code end with the following:
Load the binary integer into R1
Zero R2
Issue the LEAVENUM macro just before LTORG
 - For a floating-point result, end the code with the following:
Load F0 with the long floating-point number
Issue the LEAVEF0 macro just before LTORG
- To avoid affecting the performance of Model 204 multithreading operations, ensure that a User Language function does not invoke any kind of synchronous operating system service that could result in a wait.

Coding messages

You can add messages and codes specific to user-written functions by adding EDEF message lines to the MSGU module. These are referenced in your functions by invoking the ERROR macro.

To code and activate messages and return codes:

1. Add EDEF statements to MSGU as needed in place of the dummy EDEF in the distributed module.
2. Add ERROR macros to your functions, referring to messages in MSGU by message number as follows:

```
ERROR errnum,OPT=USER
```

3. Reassemble MSGU and FUNU.
4. Relink the Model 204 modules (for example BATCH204, ONLINE) in which MSGU and FUNU are included.

EDEF statements in MSGU

Each EDEF statement defines a message.

EDEF statement syntax

```
EDEF msgno,aaaa,class,batchrc,ssss,text
```

Where

- *msgno* denotes message number
- *aaaa* denotes action bits controlling output, snap generation, register pointers, and so on.
- *class* (E, I, or P) specifies batch and Online return code requirements.
- *batchrc* specifies batch return code.
- *ssss* denotes snap bits controlling contents of snaps and dumps.
- *text* contains message text, which can contain %variables passed by invocation of the ERROR macro.

For complete details on these EDEF arguments, refer to the internal documentation in the MSGU listing.

Encoding/Decoding facility

The Model 204 User Language \$CODE and \$DECODE functions provide the capacity to operate with both coded and string values of fields. These functions are described in the *Model 204 User Language Manual*.

CDTB module

Define code tables in the CDTB module, which you must include in any load module needed to support the \$CODE and \$DECODE functions (usually with BATCH204 and ONLINE configurations).

CDTB is an assembly language program consisting of:

- Three macros used to define each table distributed in source code form
- Comments explaining the macros
- CSECT statement
- Three sample tables
- END statement

The macros contained in the CDTB module are:

- CODETABL *id*, *length*

Where

- *id* is a 1- to 8-character string, not enclosed in quotation marks, used to

identify the table.

- *length* is the length of the longest code found in the table.

The CODETABL macro builds a table header that consists of a pointer to the next table, the table identifier, the number of entries in the table, and the length of the longest code. Each table defined must specify one CODETABL macro.

- `CODE code, 'string'`

Where

- *code* is the code not enclosed in quotation marks and not longer than the length specified in the previous CODETABL macro.
- *string* is the string associated with the code and is enclosed in quotation marks. The string can have up to 255 characters.

The CODE macro builds one entry that consists of the code, the string length, and the string value. The CODE macro must be used individually for each entry in the table.

- ENDTAB, which requires no operands and indicates the end of the current table. One ENDTAB macro is required for each table. ENDTAB completes the definition of the table by:
 - Returning to the header created by the previous CODETABL macro
 - Updating the information about the pointer to the next table
 - Updating the number of entries in the table just defined
 - Setting up information to start the next table or to mark the end of all tables

Customizing the sample translation tables

To modify the distributed sample translation tables (refer also to the *Model 204 Installation Guides*):

1. List CDTB with its sequence numbers.
2. Replace the sample tables by using an appropriate operating system utility program.
3. Assemble the program.
4. Link the object modules into the appropriate Model 204 load modules.

Modifying translation tables

To make additions, deletions, or changes to translation tables (refer also to the *Model 204 Installation Guides*):

1. Modify the source code.
2. Assemble the source code.

3. Link-edit the new CDTB module into the appropriate Model 204 load modules.

Sizing the CDTB module

The size of the CDTB module depends on the tables defined. If tables are very large, CDTB might require a significant amount of storage, resulting in an adjustment to the REGION size.

Each table entry requires:

Number of bytes in the longest code + 1 (for string length) + the number of bytes in the string + 1

32 bytes per table is required overhead for header information.

Converting user-written functions

User-written \$functions may need to be modified if you are migrating to 31-bit or multi-processing environment. While Rocket Software generally discourages customers from coding \$functions that have extensive dependencies on Model 204 internal data structures and algorithms, some customers have found it necessary to do so. These conversion notes describe the changes required for the conversion of most \$functions.

Note: These sections do not provide complete descriptions of all internal data structure and algorithm changes that have been made to Model 204; nor are they to be construed as official sanction on Rocket Software's part of the use of any specific interfaces or of the dependency on any specific data structure or algorithm. **All such interfaces, data structures, and algorithms remain subject to change without notice.**

- The section "Additional requirements for systems using 31-bit architecture" applies to all sites that use 31-bit architecture.
- The section "Additional requirements for MP/204 (multiprocessing)" applies to users of MP/204 (multiprocessing).

Coding requirements for all operating systems

Observe the following coding requirements for all operating systems:

- Model 204 runs in 64-bit mode and requires all registers to be saved and restored using the grande versions of the assembly instructions STM (Store Multiple) and LM (Load Multiple). All user-written functions must be reviewed and modified to use STMG and LMG instructions.
- Change \$functions that refer to KX or KY as follows:
 - Remove X,Y arguments from the KOMMS macro.
 - Code the new FSA macro with the same X,Y arguments that were

present on the KOMMS macro.

- Modify references to fields in KX or KY to use a different base register from the KOMM base register as follows:

```
L      reg, KJFLKX
USING  KX, reg
refer to KX or KY fields
DROP   reg
```

- From the CCALL MOVEIN routine, remove CCALL. This places the code inline. By expanding the code, efficiency is improved. The input and output are the same as before. However, you might have to add pool statements to the macro, since the offset points to a pool.
- Some KOMM fields have been moved from KA-KD to KZ. While user-written \$functions should not generally reference these fields, those that do need to address KZ as follows:

```
L      reg, KBPKZ
USING  KZ, reg
refer to KZ fields
DROP   reg
```

- The CSAVE or IOSAVE macros replace the SETAMODE macro in the MACLIB.
- The AMODE parameter of the ENTER macro is ignored. The CCALL, ENTER, and KARTN sequences no longer cause switches in addressing mode.
- The XAMODE macros are now obsolete and have been deleted from Rocket Software-delivered MACLIBs. Consequently, it is no longer necessary to maintain XAMODE\$U.
- Reassemble FUNU and FUNUUG.

Additional requirements for systems using 31-bit architecture

All code receives control in 31-bit mode. See *IBM z/OS/XA Principles of Operation* for differences between 24-bit and 31-bit mode operation:

- Nearly all Model 204 data structures (servers and PCBs, for example) are in 31-bit storage. Pointers to such data structures must be 31-bit addresses. Three-byte address constants, for example AL3(xyz), cannot be used as pointers.
- Code that explicitly manipulates Model 204 4-byte string codes must be modified to remove the high-order byte from the string code prior to adding KUPVT or KUPST.
- If you invoke a z/OS service that must receive control in 24-bit mode (such as BSAM) use CSAVE or IOSAVE with the AMODE=24 parameter. Code

CRSTR or IORSTR with the AMODE=31 parameter. You cannot reference KOMM or the pushdown list between AMODE=24 and AMODE=31 brackets.

- The NOARG parameter (described on page 238) is required for ARG statements in 31-bit mode.
- You cannot use the CCALL macro in 24-bit mode.
- For users of the multiple KOMM feature:
 - If the \$function switches to z/OS register conventions, RC might at some point no longer be equal to KOMM. Use the following code to locate KOMM:


```
CURROSW REG=x
LLGF Y,OSWPUK-OSW(,x)
```
 - In ESA systems, each user's KOMM is allocated in 31-bit storage. (See the discussion of MP system parameters in Chapter 18: "Performance Monitoring and Tuning").

Additional requirements for MP/204 (multiprocessing)

- The FN macro has an MP argument, which defaults to "NO". Other legal values are "OK" and "YES". When the FN macro specifies MP=NO, the \$function is guaranteed to execute in the maintask.
- When a \$function is invoked that specifies MP=OK, that \$function might execute in one of the MP subtasks, depending on whether Model 204 is running in parallel a section of code that contains it. If so, the \$function is eligible to execute in a subtask.
- When a \$function is invoked that specifies MP=YES, that \$function executes in one of the MP subtasks, regardless of whether the section of code within which it is contained is running in parallel. Usually, only \$functions with relatively long pathlength should specify MP=YES, because the cost of switching from the maintask to a subtask can be in the thousands of instructions. Because the exact cost of switching tasks depends on how busy the subtasks and the maintask are, measurements should be made in marginal circumstances.
- \$functions that are eligible for subtask execution actually execute on one of the subtasks under either of the following conditions:
 - SCHDOPT=X '02' is specified.
 - The section of code in which the \$function is coded is executed at a time when other users are on the scheduler's READY queue.
- \$functions that specify MP=OK or MP=YES
 - Are not guaranteed to always execute in the same subtask when NMP-SUBS is greater than 1.

- Must be re-entrant.
- Must not use any internal Model 204 interfaces that require maintask execution, for example, APUT, DKIW, and CENQxxx.
- MP=OK and MP=YES have no effect when the MP feature is not in use (NMPSUBS=0).

Part II

Managing Security

[Redacted]

[Redacted]

[Redacted]

This part describes the Model 204 system manager's role in security.

10

Storing Security Information (CCASTAT)

In this chapter

- Overview
- Creating CCASTAT
- Using CCASTAT
- Login security
- Overview of the Password Expiration feature
- Understanding the ZCTLTAB utility
- System manager's responsibilities

Overview

CCASTAT is a sequential data set containing a password table of user and file security information in an encrypted format. All Model 204 security features depend upon the existence of CCASTAT.

This chapter describes how to create and use CCASTAT.

For additional information about security and modification of CCASTAT, see Chapter 11.

Creating CCASTAT

Before you can turn on any security options, you must create a prototype password table using the utility program ZBLDTAB.

Note: When installing external security interfaces with Model 204, you must still define a CCASTAT file that contains at least one entry.

z/OS procedures

Use the following z/OS JCL to create CCASTAT:

```
//EXEC      PGM=ZBLDTAB,REGION=50K
//STEPLIB   DD  DSN=M204.LOADLIB,DISP=SHR
//CCASTAT   DD  DSN=M204.CCASTAT,UNIT=3330,
//           SPACE=(TRK,(2,1)),DISP=(NEW,CATLG),
//           DCB=BLKSIZE=13030
```

- You can specify any direct access unit and any BLKSIZE. If no BLKSIZE is specified, ZBLDTAB assigns a default BLKSIZE of 6302 bytes.
- Each password table entry that does not have terminal security requires 26 bytes. (For more information, see the section “Terminal security” on page 276.)

z/VM procedures

When running under z/VM/CMS, the ZBLDTAB utility program initializes a Model 204 password table. The format of the ZBLDTAB command is:

```
ZBLDTAB [datasetname] [filename filetype] filemode
```

Where

- *datasetname* specifies the name of the password table data set on a variable-format disk, with the qualifiers separated by blanks. If the specified disk is variable-format and if no *datasetname* operand is specified, it is presumed that the name of the password table data set is M204.CCASTAT.
- *filename* and *filetype* signify the name and type of the password table file on a CMS-format disk.

z/VSE procedures

The z/VSE JCL statements required to execute ZBLDTAB are as follows:

```
// JOB ZBLDTAB BUILD MODEL 204 SECURITY FILE
// DLBL M204LIB,'M204.PROD.LIBRARY'
// EXTENT SYSnnn,...
// LIBDEF PHASE,SEARCH=M204LIB.V411
// DLBL CCASTAT,'security file-id',0,SD
// EXTENT SYS001,balance of extent information
// ASSGN SYS001,X'cuu'
// EXEC ZBLDTAB
/*
/ &
```

- The amount of disk storage must be sufficient to contain two Model 204 pages, 6184 bytes each.
- The minimum allocation for FBA devices is 26 blocks.
- You must provide the full EXTENT information for the file for the execution of ONLINE or BATCH204, if you are updating the password table with the LOGCTL command.
- You must turn on the UPSI switch 3 (SYSOPT=16) to activate Model 204 security facilities:

```
UPSI 00010000
```

Using CCASTAT

Data set definitions (z/OS, z/VSE, z/VM)

Runs that use security options require the following DD statement in their z/OS JCL:

```
//CCASTAT DD DSN=M204.CCASTAT,DISP=SHR
```

Use the following statements for z/VSE (see z/VSE considerations below):

```
// DLBL CCASTAT,'M204.CCASTAT',0,SD
// EXTENT SYSnnn, (balance of extent info)
// ASSIGN SYSnnn, X'cuu'
```

z/VM/CMS requires a FILEDEF. For example:

```
FILEDEF CCASTAT G DSN M204 CCASTAT
```

Encryption prevents security breaches of the password table itself. Decryption occurs when a disk copy of the table is read into main memory. If Model 204 cannot decrypt the table, it produces a message stating that the password table has been corrupted.

z/VSE security considerations

Under z/VSE, the password table is a standard sequential disk file. When a LOGCTL command is entered that requires the CCASTAT data set to be updated, the file is closed for input, opened for output, and written to disk.

If the file has not expired, the operating system issues a message indicating that the existing file is going to be overlaid as a result of the operation:

```
EQUAL FILE IN VTOC CCASTAT
```

To complete the update of the password table, the operator must respond to the message. If a response other than DELETE is given, the operating system

attempts to cancel the ONLINE job stream. A message indicating that a STXIT has been trapped is issued.

To circumvent the STXIT message and the attendant operator response, you can create the CCASTAT data set with a retention period of zero, as in the DLBL example above. However, the resulting file might be vulnerable to inadvertent erasure by other applications without notice.

If CCASTAT is given a zero retention period, use the ALLOCATE utility to allocate two dummy data sets with a retention period of 99/365. Allocate the first data set to immediately precede CCASTAT and allocate the second to immediately follow it.

Login security

Login security limits access to Model 204 by requiring individual users to enter a user ID and a valid password when logging in.

- If users are required to log in, every user must enter a valid ID after the LOGIN or LOGON command, followed by a valid password, if one is specified for the user ID. The ID and password are used by Model 204 to determine authorization to use Model 204 and the type of operations available to the user.
- If users have the option of logging in to the system and a password table is available, user privileges are granted in accordance with the entries in the password table.

The password requirements and restrictions are the same whether a user resets their password at login or it is reset via the LOGCTL command.

- If a user does not issue a LOGIN or LOGON command, Model 204 assumes that the user has superuser privileges.

Login implementation

To implement login security, include the following in the JCL:

- Setting of 16 on the SYSOPT parameter.
- CCASTAT statement in the JCL for the run that points to a previously created password table data set. The discussion of the password table data set starts in “Password table data set” on page 253.
- Specification of the number of consecutive times a login can fail (LOGTRY parameter) before Model 204 takes a specified action (LOGFAIL parameter).

Set LOGTRY and LOGFAIL on User 0's parameter line. You can reset LOGTRY during a Model 204 run. For more information about LOGTRY and LOGFAIL, see the *Rocket Model 204 Parameter and Command Reference*.

Login delays

In addition to the LOGTRY and LOGFAIL parameter options, an additional, automatic security feature is invoked for threads on which users make repeated attempts to login and fail due to invalid ID-password combinations.

When login is required, if an invalid user ID and password combination is supplied, an internal count used in the LOGTRY comparison is incremented. On the fourth-failed login attempt, the login process for this thread begins to slow down. The error count is used as the basis for a calculation that increases according to the function $N*(N-1)$, where N is the number of failed login attempts.

A valid user ID and password combination immediately clears the error count and eliminates the delay. During login service delays, Model 204 also drops the login service priority level from 255 to 0, and marks the offending thread as pending login due to errors.

LOGFAIL actions still occur. The user can be bumped by a user with operator or system manager privileges. Even if a bump is performed, the failing error count is not reset until a valid login on that thread occurs. Other Model 204 users and Model 204 itself are not affected by the time delay on a failed login, other than the effect that the pending login has by tying up a thread.

To monitor the system for unsuccessful logins use the MONITOR, LOGWHO, and STATUS commands. A wait type (WT) of 23 from the MONITOR command indicates that a login for the thread is pending; flags (FLGS) indicate that the user is swappable and bumpable.

Password table data set

The password table contains one entry for each user allowed to log in to Model 204. Each entry contains the following information:

- User ID
- Password associated with the ID
- Privilege byte (summed), as shown in Table 10-1, indicating the privileges and access rights available to the user after a successful login
- Priority setting indicating the priority at which the user is allocated system resources

Table 10-1. Privilege byte settings

Setting	Meaning
X '80'	Superuser privileges. The user can create files (CREATE).
X '40'	System administrator privileges. The user can issue privileged commands (MONITOR, PRIORITY, WARN, and others).

Table 10-1. Privilege byte settings

Setting	Meaning
X '20'	The user can change the file password when it is used to open a file.
X '10'	The user can change the login password when logging in to the system.
X '08'	System manager privileges. The user can issue privileged commands (LOGCTL, DUMPG, IFAMDRAIN and others).
X '04'	The user can override record security.

The password table is stored on disk in an encrypted form to prevent unauthorized users from examining its contents. Password table decryption occurs whenever a disk copy of the table is read into main memory. If Model 204 cannot decrypt the table, a message is issued stating that the password table is corrupted.

Because the information in the password table is in coded form, it is relatively protected from unauthorized access. If more protection is desired, consider IBM utility security on the CCASTAT data set. Model 204 must be given update privileges for any additional security.

Password table maintenance

You can add, delete, or change entries in the password table using the LOGCTL command. The following sections explain how to:

- Perform these maintenance functions
- Process password table updates
- Back up the password table

One of the first maintenance actions is to add a new user ID that includes system manager privileges. Then modify the SUPERKLUGE entry by changing the password and privileges, or by deleting it entirely.

Issue the LOGLST command to obtain a listing of all login user IDs with associated privileges, but not the passwords.

Refer to “z/VSE procedures” on page 250 for instructions on creating CCASTAT and z/VSE-specific considerations.

For details about LOGCTL and LOGLST, refer to the *Model 204 Parameter and Command Reference*.

Backing up the password table

Make backup copies of CCASTAT regularly by using an IBM utility or maintaining a card deck that consists of all the LOGCTL A commands executed

to date. If you use a card deck, you can reconstruct CCASTAT by running the ZBLDTAB utility and BATCH204.

Adding entries

To add new login passwords, user privileges, and priorities for specified user IDs:

1. Use the LOGADD parameter on User 0's parameter line to specify the number of slots for new entries to be added to the password table during a run.
2. Issue the LOGCTL A command.

If the specified user ID is already in the table, or if no slots are available, you receive a message.

For syntax and examples of LOGADD and LOGCTL, see the *Rocket Model 204 Parameter and Command Reference*.

By specifying non-numeric user IDs, you can avoid the ambiguity that can result in commands that accept user IDs or user numbers as arguments.

Deleting entries

To delete login password table entries, use the LOGCTL D command.

If the user ID does not exist in the password table, Model 204 returns a message stating that the specified entry was not found. If the user ID is found, the password table entry is deleted and the space can be reused for a password table addition during the run.

Changing entries

To change login password table entries, use the LOGCTL C command:

- The user ID must already exist in the password table. If it does not, Model 204 issues a message indicating that the entry was not found.
- If an entry is omitted, the corresponding entry already in the password table is preserved.
- Delimiters are required as place holders if an entry is not changed. For example:

<code>,X'80',NONE</code>	Change privileges and priority
<code>,,HIGH</code>	Change priority

A sample dialogue using LOGCTL

The following dialogue illustrates the use of LOGCTL in adding, changing, and deleting user IDs:

LOGCTL A WASHBURN

*** M204.0374: ENTER PASSWORD, PRIVILEGES, PRIORITY

SESAME,X'80',HIGH

*** M204.0379: ENTER TERMINAL LIST, ALL, NONE, ADD, DEL, OR
RETURN

3,4,7,21

LOGCTL C ABARTH

*** M204.0374: ENTER PASSWORD, PRIVILEGES, PRIORITY

*** M204.0379: ENTER TERMINAL, ALL, NONE, ADD, DEL, OR RETURN

LOGCTL D SMITH

Before making any change permanent, Model 204 displays the new password table entry for an add or change and the old entry for a delete. If the display is interrupted with an ATTENTION, the change is not made.

Processing login password table updates

The password table can be updated by concurrently running copies of Model 204 without preventing a subsequent update to the same table. In order for concurrently running copies of Model 204 to update the password table, the in-core version of the table must have sufficient core storage allocated to it by the run. The amount of core storage allocated is determined by:

- Size of CCASTAT when Model 204 was initialized
- Value of the LOGADD parameter

Updates that exceed allocated storage

A password table update can enlarge the disk version, which might make the disk version too large to fit into the core storage allocated for the table by a concurrently running job that attempts a subsequent update to the table. For example:

1. Job A and Job B are concurrently running copies of Model 204 using the same password table data set. Job A has LOGADD set to 7; Job B has it set to 5.
2. Job A updates the password table, adding six passwords.
3. Job B cannot update because not enough storage to hold the new disk version of the password table was allocated.

In this situation, if the run with insufficient storage attempts to update the password table, it receives these messages:

```
*** M204.0312: DISK VERSION OF PASSWORD TABLE TOO LARGE TO
READ INTO ALLOCATED STORAGE
```

```
*** M204.0343: CHANGE APPLIES ONLY TO THIS RUN; UPDATES TO
CCASTAT NOT ALLOWED
```

To avoid having too little storage allocated for the password table, all jobs that update the same password table should specify the same value for LOGADD. However, if the password table is updated by a job before other jobs that run concurrently are initialized, the problem of insufficient storage can still arise.

For example:

1. Job A initializes with a LOGADD value of 5.
2. Job A updates the password table, adding three passwords.
3. Job B, using the same password table, initializes with a LOGADD value of 5.
4. Job B updates the password table, adding three passwords.
5. Job A does not have enough storage even though both jobs specified the same value for LOGADD. Job B initialized after Job A updated the password table and, as a result, Job B was allocated a greater amount of core storage during initialization.

Multiple copies of Model 204

A copy of Model 204 running concurrently with other copies does not automatically have access to the updates made by the other copies. For example:

1. Job A and Job B are concurrently running copies of Model 204 that use the same password table data set.
2. Job B updates the password table.
3. Job A does not have access to Job B's updates at this point.
4. Job A updates the password table.
5. When Job A updates, Model 204 reads a current copy of the disk version of the password table into core. Because this copy was made after Job B's updates, it contains Job B's updates; therefore, Job A has access to Job B's updates. However, Job B does not have access to Job A's updates. Job B must update the password table again to get a current copy of the disk version of the password table data set.

Overview of the Password Expiration feature

The Password Expiration feature enhances the Model 204 CCASTAT security feature to manage password expiration and includes the changes described in this section.

Tracking the days a password is valid

The CCASTAT file saves password expiration and invalid logon count data for each user ID.

- PWDEXP is a view-only parameter that reflects the number of days a user can login using the same password. The password expires when the period of time elapses since the password was last changed.
- PWDPURGE is a view-only parameter that reflects the number of days an expired user ID is held in suspension in the CCASTAT file awaiting a new password from the system manager before the user ID is deleted.
- PWDWARN is a view-only parameter that reflects the number of days prior to password expiration that the user will receive the following warning at login:

```
***M204.2634: YOUR PASSWORD WILL EXPIRE IN n DAYS
```

For each user login ID requested, the LOGLST command displays:

- Password expiration date
- Password purge date
- The number of unsuccessful login attempts made

Revoking passwords and suspending user IDs

Model 204 immediately revokes a password after three, sequential, unsuccessful login attempts. A successful login after the first or second failure resets the error counter to zero for that user ID.

When a password is revoked, the user ID is held in suspension in the CCASTAT file and cannot login to Model 204. The system manager may reinstate that user ID by changing the password before the expiration date and the purge date have passed.

When the number of failed login attempts exceeds the number specified by the LOGTRY parameter, the action specified by the LOGFAIL parameter is taken. This is independent of the Password Security feature. If LOGTRY is set to 1 and LOGFAIL is set to 2, for example, the user's thread is deactivated on the second consecutive login attempt, but the user's password is not revoked until after three consecutive failed login attempts.

Deleting CCASTAT entries

An entry in CCASTAT is deleted when the password has expired and the grace period specified by the PWDPURGE parameter has elapsed. The system manager must create a new entry for that user ID.

- If the Online is running continuously from before midnight till the time a user logs in, the first user to login after midnight triggers the purge of all expired passwords that passed the grace period. A user who logged in before midnight does not trigger the purge; the trigger is a new login after midnight.
- If the Online has not been running continuously between midnight and an expired-and-past-the-grace-period user ID attempts to logon, only that user ID is deleted.

Whenever a user ID and password are deleted from the CCASTAT file, the following message is sent to the audit trail:

```
*** M204.2636: USER username DELETED FROM PASSWORD TABLE:
PASSWORD EXPIRED
```

Defining a password

A new password must differ from the current and previous password for that user. A password cannot match the value of the corresponding user ID. See “LOGCTL: Modifying user ID entries in the password table” in the *Model 204 Parameter and Command Reference*.

When a password is added or changed, confirmation is requested with the following prompt:

```
***M204.2633: RE-ENTER NEW PASSWORD
```

The system manager can specify passwords that do not expire using the NOEXPIRE keyword in the LOGCTL A and LOGCTL C commands.

FILE and GROUP passwords

Password security rules apply to only LOGON passwords.

Create a backup copy of CCASTAT

The CCASTAT file used as a testing version should *not* be your production version. Before testing begins, put a backup copy of your CCASTAT file in a secure place. You and Technical Support might need the unconverted version of your file to diagnose a problem.

Capturing diagnostic data

Create a process by which you can capture and deliver debugging materials, such as audit trails, snaps, and dumps to Rocket Software in the event that Technical Support needs to help you diagnose problems during testing.

Increase in CCASTAT

The Password Expiration feature requires an additional eight bytes per user ID record in the password table. The increase in size is required only for user entries, not file or group entries. You would only have to increase the size of the CCASTAT file, therefore, if the file is almost full or you have a very large number of user entries. The increase might be necessary depending upon the current size of CCASTAT.

Understanding the ZCTLTAB utility

The ZCTLTAB utility is a dual purpose job that can be run to read an existing CCASTAT and create a new one pointed to by ddname in JCL NEWSTAT. Also, the ZCTLTAB JCL can be modified, removing the NEWSTAT, and just be used to change values of WARN, PURGE, and/or EXP in an already existing converted CCASTAT.

Updating CCASTAT using the Password Expiration feature

When you run the ZCTLTAB utility against the original unconverted CCASTAT file, it creates a copy of the original file and installs the Password Expiration feature into the new file. The new file is no longer compatible with earlier versions of Model 204. Therefore, Rocket Software strongly recommends that you save the original unconverted CCASTAT file. Also, use this new CCASTAT file with the Password Expiration feature in testing until you are completely satisfied that it is ready for production work.

After you have initially installed the Password Expiration feature in a CCASTAT file, you may run the ZCTLTAB utility against this file to update the password expiration parameters. ZCTLTAB will detect that CCASTAT has already been converted and will not create a new file, but will update the parameters in the existing file. When running with a converted CCASTAT file, the NEWSTAT data set is not required.

The first time a job runs with a CCASTAT file converted by ZCTLTAB, the creation date for all existing user IDs are reset to the current date.

The Online changes the creation date of a user entry in the CCASTAT file that is enabled for the Password Expiration feature whenever you change the password for that user.

Setting the security parameters

The ZCTLTAB utility uses the following input parameters:

- EXP to set the number of days until the password expires.
- PURGE to set the number of days until an expired password is deleted from CCASTAT.
- WARN to set the number of days prior to expiration to start warning the user.

Displaying the security parameters

These values are set within the CCASTAT file. Use the following VIEW command to view the:

- EXP value

```
VIEW PWDEXP
```

- PURGE value

```
VIEW PWDPURGE
```

- WARN value

```
VIEW PWDWARN
```

Changing the values of the security parameters

To change the parameter values, your site simply runs ZCTLTAB with new values for EXP, PURGE, and/or WARN. If you change only one parameter value, the other parameter values remain as they were. Running the ZCTLTAB utility a second time (or more times) does not reset the creation dates of the entries, as the first run did.

ZCTLTAB condition codes

The following condition codes are feedback from the ZCTLTAB utility.

Condition code	Meaning
0	Success
4	Unable to open a file
8	Invalid parameters
12	The version of CCASTAT you used is too old
16	I/O error occurred when reading or writing a file
20	Unable to obtain storage (z/VSE only)

No changes are made to the CCASTAT file if the condition code is other than 0.

The ZCTLTAB utility writes a message to the CCAOUT data set that explains the condition code or an error. See “ZCTLTAB messages” in the *Rocket Model 204 Messages Manual*.

Running ZCTLTAB to update CCASTAT

The ZCTLTAB utility is unloaded during installation and is used to convert and maintain the CCASTAT file. You can reset the Password Expiration parameters EXP, PURGE, and WARN. This updates your current CCASTAT file and the CCAOUT file to capture the Password Expiration error messages.

Sample JCL for z/OS ZCTLTAB utility

```
//ZCTLTAB JOB (system manager),'system manager ',CLASS=A,MSGCLASS=A
//*****
//*      Job: ZCTLTAB      created on: 22 NOV 2009  AT: 16:18:10
//*      OPTIONAL Job to create/update CCASTAT with expiration passwords
//*      Updates password expiration parameters in CCASTAT
//*      1. Initializes the password expiration feature in CCASTAT if
//*          it was not previously converted
//*      2. Changes password expiration parameters in CCASTAT
//*          that have already been converted
//*      Include the NEWSTAT DD card only when running ZCTLTAB on a
//*      CCASTAT file for the first time.
//*      Notes:
//*      Add jobcard and modify parameters on EXEC card to set days till
//*          Expiration, start of Warning messages, and length of grace
//*          period before the password is Purged. Change DSNs for
//*          LOADLIB and the target CCASTAT. If NEWSTAT is used, modify
//*          new data set parameters as needed.
//*****
//* Notes:
//* May be run to convert and maintain CCASTAT to support
//* expiring passwords. Read Install Guide and job comments
//*****
//CONVERT PROC SYSINDEX='M204',RLSE=V610n,
//          DISK=nnnn,VOLSER=nnnnnn
//ZCTLTAB  EXEC  PGM=ZCTLTAB,PARM='EXP=20,WARN=19,PURGE=255'
//STEPLIB  DD  DSN=M204.V610n.LOADLIB,
//          DISP=SHR
//CCASTAT  DD  DSN=&SYSINDEX..&RLSE..CCASTAT,DISP=SHR
//NEWSTAT  DD  DSN=&SYSINDEX..&RLSE..NEW.CCASTAT,
//          DISP=(,CATLG,DELETE),
//          SPACE=(TRK,25),
//          UNIT=&DISK,
//          VOL=SER=&VOLSER
//CCAOUT   DD  SYSOUT=*
//          PEND
//SECURE   EXEC  CONVERT
//
/*      End of ZCTLTAB *****
```

Sample JCL for z/VSE ZCTLTAB utility

```
// JOB ZCTLTAB
* *****
*                               Computer Corporation of America
*                               6/09
* *****
* ZCTLTAB creates and maintains the converted CCASTAT: (NEW/CCASTAT)*
* *****
/* Job: ZCTLTAB                      Created: Sept. 2009          *
/* 1. Creates a new CCASTAT from a standard CCASTAT, copying in *
/* security information and setting initial expiration values.*
/* 2. Changes password expiration parameters in a CCASTAT      *
/* that has already been converted.                             *
/* Include the NEWSTAT DD card only when creating a converted  *
/* CCASTAT file. To maintain a converted CCASTAT, remove NEWSTAT.*
/* *****
/* Notes:                                                         *
/* Add jobcard and modify parameters on EXEC card to set        *
/* n_days till Expiration, start of Warning messages,          *
/* and length of grace period before the password is Purged.   *
/* Change DSNs for LOADLIB and the target CCASTAT.             *
/* If NEWSTAT is used, modify new data set parameters as needed.*
/*                                                              *
/* *****
// EXEC PROC=M204JCL          DLBL for JCL sublibrary
// LIBDEF PROC,SEARCH=M204LIB.V610n
// EXEC PROC=M204V610          DLBL for M204 library
// LIBDEF PHASE,SEARCH=M204LIB.V610n
// DLBL CCASTAT,'M204.CCASTAT',99/366,SD
// EXTENT SYSnnn,volser,,,start,length
/* NEWSTAT DLBL required only if creating new converted CCASTAT
/* To update an existing CCASTAT supporting expiring passwords,
/* remove or comment out the NEWSTAT JCL
// DLBL NEWSTAT,'M204.EXPIRE.CCASTAT',99/366,SD
// EXTENT SYSnnn,volser,,,start,length
// EXEC ZCTLTAB,SIZE=AUTO,PARM='EXP=n_days,WARN=n_days,PURGE=n_days'
/*
/ &
```

Using ZCTLTAB to modify CCASTAT

When you run ZCTLTAB to update an already-converted CCASTAT data set, follow the comments in ZCTLTAB to remove references to the NEWSTAT data set.

Note: If this data set is not omitted in subsequent execution, an empty reference file may be created.

Setting EXP, PURGE, and WARN in the ZCTLTAB utility

The valid values of EXP, PURGE, and WARN are from 0 to 255.

In the initial run of the ZCTLTAB utility to convert the CCASTAT file, you must explicitly set the EXP parameter.

Setting EXP=0 results in the expiration of all passwords in the CCASTAT file, unless they have the NOEXPIRE option. Only user IDs with the NOEXPIRE option will be able to login.

Setting EXP=0 is valid only if WARN=0 is also set or currently exists in a converted CCASTAT.

Viewing the ZCTLTAB parameters

If you issue a VIEW command for PWDEXP, PWDPURGE, and PWDWARN and they display a value of -1, CCASTAT has not been converted, and Model 204 logins behave as they did prior to installing the Password Expiration feature.

System manager's responsibilities

In a security emergency, the system manager can change the expiration of all passwords for all jobs accessing a CCASTAT file without bringing those jobs down by running the ZCTLTAB utility.

For a running job to read the updated copy of CCASTAT, the job must make a change to CCASTAT in one of the following ways:

- LOGCTL A command
- LOGCTL C command
- A valid user ID attempts to log in; the user ID is in the password table, but did not enter the correct password.
- The job notices that midnight has passed and finds user IDs to delete after a login is attempted. Model 204 notices when midnight passes and deletes expired passwords when the first user of the day attempts to logon.

PWDEXP, PWDPURGE, and PWDWARN are view-only parameters in Online and Batch204 runs. The only way to change the values is with the ZCTLTAB utility designated by EXP, PURGE, and WARN. Every job that runs with that CCASTAT file from then on will use those values.

Note: Any user who is authorized to submit the ZCTLTAB job can reset the EXP, PURGE, and WARN values. Rocket Software recommends storing this job in a secured data set.

Supporting password expiration at your site

The ZCTLTAB utility checks whether the EXP utility parameter is not zero when the WARN utility parameter is not zero. If the EXP parameter is set to zero when WARN is not set to zero, a Return Code of eight (8), meaning invalid parameter, is invoked.

CCASTAT may be shared between multiple Online and Batch jobs, as in previous versions.

If the ZCTLTAB utility is not run, then the value -1 is displayed for the PWDEXP, PWDPURGE, and PWDWARN parameters, when you issue a VIEW command against them. There is no password expiration and no warning of impending expiration.

Deleting a user ID

After a password expires, the password and user ID are deleted from the system when the PWDPURGE number of days has elapsed. The audit trail automatically records deleted user IDs.

For Onlines that do not run over midnight, a user whose password has expired past the purge period is deleted from CCASTAT when that user attempts to login. It does not occur when another user logs in.

Defining passwords

Follow the instructions for defining passwords listed in “LOGCTL: Modifying user ID entries in the password table” in the *Model 204 Parameter and Command Reference*.

- If you do not run the ZCTLTAB utility, you can apply the lesser restrictions.
- After you run the ZCTLTAB utility, you must use the more detailed restrictions.

Changing and reusing passwords

After you run the ZCTLTAB utility, when passwords are changed, they cannot be a repeat of:

- The current password
- The previous password
- The value of USERID

Waiting to change a password

Users waiting to update CCASTAT with a new password simply wait until CCASTAT is available. The wait type is 23. In previous versions of Model 204,

when a user was changing a password, then another user attempting to do the same simultaneously received the following message:

```
***M204.0370: TABLE IN USE, TRY AGAIN
```

Now bumpable, swappable waits replace the message. For updating users, the message is eliminated.

Password from a trusted environment

No password expiration verification is done for logins from trusted environments, which is the behavior of previous versions of Model 204.

CCASTAT backwards compatibility

Prior versions of Model 204 cannot run with the CCASTAT file created for V6R1.0. This backward incompatibility is present when the parameters that control password expiration have been used in a run that has Read-Write access to the CCASTAT file.

The system manager should save the original, unconverted CCASTAT file. Once a CCASTAT file is converted for this version, previous versions of Model 204 can open and look at, but not run with the converted CCASTAT. Doing so results in the message:

```
M204.0337: WRONG VERSION FOR CCASTAT
```

Managing CCASTAT messages

When you begin using the Password Expiration feature there is a greater likelihood of CCASTAT enqueue conflicts, because Model 204 is looking at CCASTAT for more information, so it holds enqueue longer. Also, because of password expirations, CCASTAT is updated more frequently.

You might see the following or other messages that reveal enqueue conflicts:

```
M204.0196: DATA SET dataset-name USED BY CCASTAT DD  
STATEMENT CURRENTLY IN USE
```

```
M204.0344: DISK VERSION OF CCASTAT CHANGED BY JOB job-  
name yy.ddd hh:mm
```


11

Establishing and Maintaining Security

In this chapter

- Overview
- File security
- Group security
- Record security
- Field-level security
- Procedure security
- Terminal security

Overview

Model 204 provides basic internal security features at all levels.

All security features are optional. Any combination of features is supported.

Chapter 10 explains how to maintain the CCASTAT data set, which contains a password table of encrypted user and file security information.

For information about Model 204 and standard commercial security interfaces, refer to the *Model 204 Security Interfaces Manual*.

File security

The file security feature limits access to particular files by requiring a password to open the file. The type of operation a particular user can perform on the file is also controlled by the file password.

OPENCTL parameter

The OPENCTL parameter, issued during creation of the file (or reset during a Model 204 run) determines whether a file is public (no password required) or is protected by one of several levels of file, group, and record security.

For OPENCTL settings that pertain to Parallel Query Option/204, see the *Rocket Model 204 Parameter and Command Reference*.

PRIVDEF parameter

Access to a file is limited by the settings on the PRIVDEF parameter. The PRIVDEF parameter summarizes the default file privileges that are assigned when a public file is opened, or when a semipublic file is opened without a password or with an invalid password. User privileges are included in the PRIVDEF specification.

Set PRIVDEF to any combination of values listed in Table 11-1 on page 270.

If you must restrict the viewing of file-related parameters, reset the default value (X'BFFF') of PRIVDEF with the RESET PRIVDEF command. Otherwise, the odd setting of PRIVDEF allows a user to view file-related parameters through one of the following display commands or functions:

- DISPLAY FILE or DISPLAY VIEW
- \$VIEW
- IFDISP or IFPERM

PRIVDEF parameter settings

Table 11-1 lists PRIVDEF settings, which you can add in any combination.

Table 11-1. PRIVDEF parameter values

Setting	User can...
X '8000'	Use privileged commands such as INITIALIZE, SECURE, and DESECURE. The user can also reset file parameters if ad hoc update privileges (X '2000') are obtained.
X '4000'	Override record security.
X '2000'	Update data with ad hoc requests or host language programs.

Table 11-1. PRIVDEF parameter values (Continued)

Setting	User can...
X '1000'	Make changes to procedures defined in the same file as the data but cannot delete them.
X '0800'	Update data with internal procedures.
X '0400'	Retrieve data with ad hoc requests or host language programs.
X '0200'	Display, echo, and copy internal procedures.
X '0100'	Retrieve data with internal procedures.
X '0080	Update data with procedures defined in a different file from the data.
X '0040'	Retrieve data with external procedures.
X '0020'	Include internal procedures.
X '0010'	Define internal procedures.
X '0008'	Delete internal procedures.
X '0001'	Access file related parameters.

File password table maintenance

The password table contains one entry for each file password. These entries consist of the following information:

- File or group name
- Password that the user must specify in order to be granted the file or group privileges
- Two-byte representation of the user privileges that is granted when the user successfully opens the file or group (see Table 11-1)
- User class for procedures
- Field-level security SELECT, READ, UPDATE, and ADD levels

Using LOGCTL to modify the file password table

The LOGCTL command allows you to add or delete file entries in the password table. For example, the following command adds a file to the password table:

```
LOGCTL A :file-name
```

You can specify an index (i), necessary to differentiate entries for future changes or deletions, when a file has more than one password.

If you change a file or group password entry and omit one or more specifications, the corresponding entries in the password table are preserved.

To list all the file entries in the password table, along with the associated privileges (but not the passwords), issue the LOGFILE command.

Sample dialogue using LOGCTL

The following dialogue illustrates the use of LOGCTL to add, change, and delete file entries in the password table:

```
LOGCTL A :CENSUS1
*** M204.0374: ENTER FILE/GROUP PASSWORD, PRIVI-
LEGES, CLASS,
                SELECT, READ, UPDATE, ADD

ACCESS, X'0900', 60, 20, 30, 30, 40
*** M204.0379: ENTER TERMINAL LIST, ALL, NONE, ADD, DEL, OR
RETURN

21, 31

LOGCTL C :TCENSUS3
***M204.0374:ENTER FILE/GROUP PASSWORD, PRIVILEGES,
CLASS,
                SELECT, READ, UPDATE, ADD

, , 70

*** M204.0379: ENTER TERMINAL LIST, ALL, NONE, ADD, DEL, OR
RETURN

LOGCTL D :XCENSUSA
```

Adding and removing security from files

To secure a file, issue the SECURE command, which ensures that a user cannot access a file illegally by running a Model 204 program with its own password table.

A special field in the password table serves as the key for securing files. When a secured file is opened, the key is compared with a copy placed in the file by the SECURE command. The file is accessed only if the two passwords match. The comparison with the key is performed even when the file is opened as part of a group.

To reverse the security placed upon a file by the SECURE command, issue the DESECURE command.

The SECURE and DESECURE commands are discussed in detail in the *Model 204 File Manager's Guide* and the *Model 204 Parameter and Command Reference*.

To change the key in the password table, issue the LOGKEY command (see the *Rocket Model 204 Parameter and Command Reference*).

Group security

The group security feature restricts access to particular file groups to certain users. When the group is created, access is limited by parameter settings of the CREATE GROUP command. A group can be classified as:

Classification	Meaning
Public	When a public group is opened, default group privileges are defined on the PRIVDEF parameter in the group definition.
Semipublic	When a semipublic group is opened without a password or with an invalid password, default group privileges are defined on the PRIVDEF parameter in the group definition.
Private	A password is required to open the group.

In addition to the file privileges discussed in the section “File security” on page 270, you can assign two additional classifications with the PRIVDEF parameter:

Classification	User can...
X '0004'	Update data via procedures from the procedure file.
X '0002'	Retrieve data via procedures from the procedure file.

Group entries have the same format as that shown for file entries earlier in this chapter.

LOGCTL and LOGGRP commands

The following considerations apply:

- Use the LOGCTL command to add or delete group entries in the password table. For example, the following command changes the entries in the password table for the group named AREA:

`LOGCTL C ,AREA`
- You can specify an index (i), necessary to differentiate entries for future changes or deletions, when a file has more than one password.
- To list all the group entries in the password table, along with the associated privileges (but not the passwords), issue the LOGGRP command.

Record security

The record security feature limits access to individual records in a file. Each user can retrieve and update only records that the user has stored in the file or

that other users have agreed to share. The existence of other records is not apparent.

The following considerations apply:

- Record security can be in effect for one or more of the files in a group, but not for the group as a whole.
- Access to a single record depends only on the record security field defined for the record's file.
- Record security cannot be used unless login security is in effect.
- To initiate record security, set the OPENCTL parameter in the CREATE command for the file and describe the special record security field in the INITIALIZE command. See the *Rocket Model 204 File Manager's Guide* for information about these commands.
- Record security can be overridden if the user has been granted the record security override privilege at login time (the X '04' bit must be set) and the user's file password also contains the record security override privilege (X '4000').
- Record security can be overridden for a file in a permanent file group if login and group privileges (X '4000') allow it.

Field-level security

The field-level security feature (FLS) controls access to the individual fields of a Model 204 file, if access to a record is allowed by previous file-level and record-level security checks.

Every field definition (created by the DEFINE FIELD command) can have a security level from 0 to 255 associated with it. DEFINE FIELD is described in the *Model 204 File Manager's Guide*.

A level of 0 implies no security for the field; 255 implies the highest security. Field access types are described in the following sections.

Scope of field-level security

Field-level security controls only explicit field references. Implicit references, such as retrieving a record security field with a FIND statement or adding a record security key value with STORE, are not controlled.

To include user field access levels to file and group passwords in the password table, use the LOGCTL command to add or delete access entries.

The way access levels are determined for files that are also members of groups is summarized in the *Model 204 File Manager's Guide*.

Reference context and user field-level security violations are discussed in the *Model 204 Parameter and Command Reference*.

Field access types

Each user is assigned access levels, ranging from 0 to 255, for each file and group opened. Access to a field is limited to any combination of the following access types:

Access type	Means the ability to...
ADD	Add new occurrences of a field, including those added by a STORE RECORD statement. ADD access lets data entry clerks or other personnel add new field occurrences or records without being able to change existing occurrences, or possibly even to examine them. Add access can also provide a user with the ability to add occurrences of the record security field without altering existing occurrences.
READ	Examine the value of a field (for example, in a User Language PRINT or assignment statement).
SELECT	Use the field in a User Language FIND statement or an IFFIND call.
UPDATE	Change the value of a previously stored occurrence of a field. UPDATE access can be granted without a corresponding READ access, which precludes updates of the form: <i>CHANGE fieldname=value1 TO value2</i>

When a user attempts to access a field in a particular way, Model 204 compares the user's access levels with the field level defined for the field. If the user's level for the desired type of access is greater than or equal to the field's FLS level, the particular type of field access is allowed.

For example, a user who has a READ level of 30 is permitted to display any field that has a READ level between 0 and 30, but cannot display a field that has a READ level of 40.

Sample dialogue using LOGCTL

The following example adds field security levels 50 (SELECT), 40 (READ), 10 (UPDATE), and 0 (ADD); no other information is changed:

```
LOGCTL C :CENSUS
*** M204.0374:ENTER FILE/GROUP PASSWORD, PRIVILEGES,
CLASS,
                SELECT,READ,UPDATE,ADD
,,,50,40,10,0
*** M204.0379: ENTER TERMINAL LIST,ALL,NONE,ADD,DEL,OR
RETURN
```

Procedure security

The procedure security feature limits access to defined procedures.

The following considerations apply:

- You can specify privileges to enable a user to manipulate a procedure (display, define, or delete the procedure).
- You can limit access to a procedure to a particular class of users by assigning a user class number to the procedure being secured.
- Certain privileges (such as SECURE and DESECURE) associated with file and group passwords indicate the user privileges that pertain to the procedures for that file or group.
- To assign user class and procedure class mappings (see the *Rocket Model 204 File Manager's Guide*) to file and group passwords, use the LOGCTL command to add the mappings to the password table.

Use LOGCTL to add, delete, or change the user class for procedures defined for a file or group.

Sample dialogue using LOGCTL

The following example adds a user class of 70 to an existing file entry; no other information is changed:

```
LOGCTL C :CENSUS
*** M204.0374: ENTER FILE/GROUP PASSWORD,
PRIVILEGES, CLASS, SELECT, READ, UPDATE, ADD

,,70
*** M204.0379: ENTER TERMINAL LIST, ALL, NONE, ADD, DEL, OR
RETURN
```

Terminal security

The terminal security feature restricts access to certain login user IDs, files, or groups to users at identified terminals by associating a list of user numbers with each login, file, or group password. During Model 204 initialization, a terminal can be assigned a particular user number according to the order of the user parameter lines and the way in which they are assigned to specific telecommunications unit numbers in the JCL.

For example, a user at a particular terminal that has a specific number can log in to a specific user ID or open a specific file or group only if the terminal number is in the terminal list associated with the password for that user ID, file, or group. If it is not, Model 204 responds as if the user entered an invalid password.

Terminal security is generally used only with hard-wired terminals (terminals on leased lines). For dialup terminals, the terminal can be connected to a number of similar telecommunications units and user parameter lines. Even though the location of a dialup terminal is fixed, its terminal number can change every time it is dialed up.

The following considerations apply:

- To insert terminal lists into the appropriate password table entries and to change existing terminal assignments, use the LOGCTL command.
- To obtain listings of the login, file, or group entries in the password table, with legal terminal numbers included, issue a LOGLST, LOGFILE, or LOGGRP command.
- To update all terminal lists at once, use the TMASKUPDATE command.
TMASKUPDATE takes no arguments. It loops through the password table, displays each entry for which ALL has not been specified, and allows you to specify a new terminal list or a blank line (no change).
- All password table entries are treated as if terminal security were in effect, even if the terminal security feature is not used:
 - Terminal lists are variable length, allowing you to add terminals even if you exceed the original number planned for.
 - If you specify ALL for a password table entry, the terminal list takes up no space beyond the basic entry.
 - A list representing NONE takes two bytes.
 - To compute the length of a list containing numbers, divide the highest specified number by eight and rounding up to a multiple of two.

Part III

Auditing and Problem Determination

[Redacted]

[Redacted]

[Redacted]

This part describes the Model 204 system manager's role in auditing and determining problems using files created by Model 204.

12

Obtaining User 0 Output (CCAPRINT)

In this chapter

- Overview
- CCAPRINT data set

Overview

The output from CCAPRINT can help in problem determination by showing parameter or other initialization errors.

This chapter explains how to store and print User 0 output using the CCAPRINT system file.

CCAPRINT data set

CCAPRINT defines a sequential output data set (usually SYSOUT=* in z/OS) that contains User 0 output, such as:

- Summary of the execute parameters, such as SYSOPT
- Version running
- User 0 parameter settings
- Number of users and servers
- Terminal output, if the commands are issued from the User 0 input stream

Initialization information about Early Warnings

At Model 204 initialization, a listing of all Early Warnings that have been applied to the module is produced. It is comparable to the output of the following command:

```
DISPLAY EW ALL
```

The output is directed to CCAPRINT in Online, Batch204, IFAM1 (if CCAPRINT has been defined), and IFAM4 jobs. A listing of all maintenance currently applied to the active load module or phase is displayed. Messages of the following type are produced in CCAPRINT, depending on the maintenance history at your site.

Examples NO EARLY WARNINGS HAVE BEEN APPLIED
EARLY WARNING 1 TO 4 APPLIED
EARLY WARNING 6 APPLIED
EARLY WARNINGS 9 TO 12 APPLIED

This listing helps Rocket Software Technical Support reduce the time required for problem resolution. You should have both CCAPRINT and CCAUDIT available when you report problems to Rocket Software Technical Support.

z/OS considerations

In a z/OS environment, the JCL must contain a DD statement for CCAPRINT. CCAPRINT is normally defined as a SYSOUT data set. Records written to CCAPRINT go to the system spool. For example:

```
//CCAPRINT DD SYSOUT=*
```

If CCAOUT, which is used for problem analysis, is allocated in your UTILJ JCL, journal records are written to CCAOUT, not CCAPRINT. See “Using UTILJ to analyze problems” on page 400 for more details.

z/VSE considerations

In a z/VSE environment, the output from CCAPRINT is printed in the order generated on the logical unit SYSLST. The unit SYSLST must be assigned.

z/VM considerations

In a z/VM environment, CCAPRINT can be defined as one of the following:

- CMS file:

```
FILEDEF CCAPRINT DISK WORK CCAPRINT A
```

- Service machine virtual printer:

```
FILEDEF CCAPRINT PRINTER
```

Information about unusual Model 204 activity is displayed on the service virtual machine console. Spooling this console and ensuring that the file is closed and examined on a regular basis may be helpful.

Note: Do not define a terminal as the output device for CCAPRINT.

If CCAOUT, which is used for problem analysis, is allocated in your UTILJ JCL, journal records are written to CCAOUT, not CCAPRINT. See “Using UTILJ to analyze problems” on page 400 for more details.

CCAPRINT data set

13

Tracking System Activity (CCAJRNL, CCAAUDIT, CCAJLOG)

In this chapter

- Overview of the journal data sets
- Using Model 204 journal files
- Creating and generating CCAJRNL
- SWITCH STREAM command
- Using the CCAJRNL data set
- Performance efficiencies using CCAJRNL
- CCAJRNL data set record layout
- Introducing Model 204 statistics
- Monitoring statistics
- Audit trail format
- Generating an audit trail
- Introducing the AUDIT204 utility
- Using the AUDIT204 utility

Overview of the journal data sets

This chapter discusses the journal data sets you can create and use in Model 204, how to manage them, and the utilities you can use to report on and analyze the Model 204 information collected. The system manager can set up the following journal data sets, which are not part of Model 204 installation.

The data sets are CCAJRNL, CCAAUDIT, and CCAJLOG. You set each up as:

- A ddname in z/OS
- A FILEDEF in z/VM
- A DLBL in z/VSE

Introducing CCAJRNL

The complete repository of Model 204 system activity and data that would be used in recovery is CCAJRNL. The CCAJRNL file collects all update information used to reconstruct the database during recovery and all other activity history, such as messages and statistics, in an unformatted, therefore unreadable, binary format.

Introducing CCAAUDIT and CCAJLOG

You can also create a CCAAUDIT and/or a CCAJLOG file to track information. CCAAUDIT and CCAJLOG are comprised of the same Model 204 audit trail data, however, differently. The activity history or audit trail is comprised of:

- Important characteristics of a run
- All error messages encountered during a run
- All communications with the operator's console
- Input lines from Online terminals
- HLI calls
- Utilization statistics
- Information specifically directed to the journal/audit trail by users

Watching your system in action: CCAAUDIT

The CCAAUDIT file, also known as the audit trail, logs information about a Model 204 run and prints the information in readable format for the terminal screen while processes are running or to paper.

Improving recovery performance: CCAJLOG

The CCAJLOG file off loads the audit trail information into itself, so that the CCAJRNL contains only the information required for RESTART recovery. Recovery performance is improved, because the audit trail information does not have to be read and rejected for recovery.

Version-specific journals

Journals created by Model 204 prior to V7R1.0 cannot be processed in V7R1.0. The reverse is also true: journals created in V7R1.0 cannot be processed by an earlier version of Model 204.

Using Model 204 journal files

Model 204 system managers and file managers use the journal files in some of the following ways:

Task	Journal used	Utility/command used
Generate an audit trail that can be viewed during processing	CCAAUDIT	
Generate an audit trail for printing out later	CCAJRNL, CCAJLOG	Audit204 utility
Regenerate a file to complete media recovery	CCAJRNL (ddname CCAGEN)	REGENERATE command
Recover Model 204 following a system failure	CCAJRNL (ddname CCARF)	RESTART command
Extract a report	CCAJRNL	UTILJ utility
Performance tuning by analyzing statistics	CCAJRNL	Audit204 utility or site written analysis program

The Model 204 system manager is expected to create the CCAJRNL file at their site for recovery purposes. The Model 204 file manager may be responsible for using the CCAJRNL file to regenerate data files.

Creating and generating CCAJRNL

A journal is generated if:

- The SYSOPT parameter includes the 128 specification
You must set BLKSIZE for CCAJRNL to not less than 6749 on up to 32K. Buffers are written to CCAJRNL either when a transaction is committed or when the buffer fills. Frequent commits may result in many short blocks/buffers being written to CCAJRNL.
- A CCAJRNL is defined with BLKSIZE set to the minimum, 6749, for

- z/OS, a DD statement
- z/VM, a FILEDEF

Statistics generated in an audit trail line are presented in “User statistics entries (Type 9)” on page 574.

The use of CCAJRNL in recovery procedures is explained in “Recovery data sets and job control” on page 371.

How to use CCAJRNL to regenerate a data set is discussed in the *Model 204 File Manager's Guide*, Chapter 21: “Media Recovery”.

CCAJRNL may take multiple extents and may span multiple volumes, since it is read forward-only. Its size cannot be limited in a way similar to the CHKPOINT data set (see “Limiting the size of CHKPOINT” on page 342), because CCAJRNL records all update activity in an Online, which may be required in a subsequent REGENERATE or RESTART process.

CCAJRNL as single data set or stream

You can define CCAJRNL as a single data set or as a stream. Using a DEFINE STREAM command, you incorporate multiple journal data sets.

- See the *Rocket Model 204 Parameter and Command Reference*, “DEFINE STREAM: Describing sequential I/O streams”.
- For parallel and ring structures, see “Example 1: Ring/parallel journal stream” on page 421.
- For GDG structure, see “Perpetual journaling for z/OS” on page 419.

SWITCH STREAM command

The SWITCH STREAM command switches a stream to the next member of a parallel, ring, concatenated, or Generation Data Group (GDG) stream. You can issue the SWITCH STREAM command for the following streams: CCAJLOG, CCAJRNL, CHKPOINT, or CHPNTS.

Using the SWITCH STREAM command

When the SWITCH STREAM command is issued, the following messages are produced:

```
M204.2712: STREAM streamname IS BEING SWITCHED
M204.2712: STREAM streamname SWITCHED VIA COMMAND
```

- When switching a journal stream, CCAJRNL or CCAJLOG, the currently active data set in that stream is closed. The next data set defined to the stream is opened when the next write to that stream is required.

- When switching a checkpoint stream CHKPOINT or CHPNTS, the currently active data set in that stream is closed after the next record is written to that data set.

If not in an extended quiesce, Rocket Software recommends that you follow a SWITCH STREAM CCAJRNL command with a CHECKPOINT command to ensure that you have a checkpoint in the current journal.

The SWITCH STREAM command is sometimes useful for CCAJLOG and rarely required for CHKPOINT or CHPNTS.

Journal block header information for SWITCH STREAM

Some journal analysis utilities require additional journal information at sites that embrace GDG streams and the SWITCH STREAM command as a means to keep their Onlines up for long periods. Model 204 includes this additional information in the header for each journal block.

To ensure that the required information is present in the first block of each journal data set created by the SWITCH STREAM command, the header has been expanded. This expansion makes the journal up to 4% larger than in previous releases.

Note: Due to these changes in journal record layouts, CCAJRNL and CHKPOINT/CHKPNT data sets are not compatible with previous releases of Model 204.

For the complete description of the header entry formats in the journal block, see “Header entries (Type 0)” on page 560.

In pre-7.4.0 versions of Rocket Model 204, each Online environment might produce a single merged journal daily. If an Online was bounced during the day, then the various journals for the day were merged into a single daily journal. Usually the daily merged journal contained records from only one run. The single merged journal could be used to automate media recovery, rather than manually assemble the recovery job journal concatenation.

Using the SWITCH STREAM command, a site can still prepare a daily merged journal. However, when the SWITCH STREAM command is used, each journal merged does not begin with the initialization of the Online. This means that no Type 12 records or M204.0061 initialization messages are included. Although Model 204 recovery, including media recovery, can successfully process these merged journals, various journal analysis utilities cannot.

Using a SWITCH STREAM CCAJRNL command during extended quiesce

In pre-7.1.0 releases, if you used the checkpoint quiesce feature, a switch to the next journal member at checkpoint quiesce could occur only when CCAJRNL was defined as a ring stream.

However, a switch at checkpoint quiesce may be desirable for all stream configurations of CCAJRNL: ring, parallel, concatenated, and GDG. The switch marks the point where the CCAJRNL data collected thus far is not needed in subsequent REGENERATE processing against files backed up during the quiesce. If file backups or dumps are taken during the checkpoint quiesce, only CCAJRNL data collected after checkpoint quiesce is useful for REGENERATE processing against those files.

A checkpoint is automatically taken if you issue a SWITCH STREAM CCAJRNL command while you are in extended quiesce, as shown in Table 13-1.

Table 13-1. SWITCH command within extended quiesce

Step	Command issued	Purpose
1	CHECKPOINT SET EXTENDED QUIESCE	Enables extended quiesce
2	CHECKPOINT or automated checkpoint	The next checkpoint, automated or command initiated, begins the extended quiesce When the checkpoint is successful, the extended quiesce is entered. While in extended quiesce, you cannot issue a CHECKPOINT command. However, you can issue a SWITCH STREAM command, such as the following:
3	SWITCH STREAM CCAJRNL	When the checkpoint is successful, extended quiesce processing can begin for backups, SnapShots, or any activity that does not involve updating.
4	CHECKPOINT END EXTENDED QUIESCE	Concludes extended quiesce. Should recovery be required due to a failure during extended quiesce, journals created prior to the SWITCH STREAM command will not be required, as the last checkpoint resides in the current CCAJRNL member.

This automated checkpoint functionality of SWITCH STREAM CCAJRNL applies only during checkpoint quiesce. The automated checkpoint functionality is not supported for CCAJLOG, CHKPOINT, or CHKPNTS.

SWITCH STREAM limitations

In order for a stream to be switched, there must be a target data set to switch to. If there is no target data set, the following message is issued:

```
M204.2712: MEMBER membername IS INELIGIBLE FOR SWITCHING
```

Consider the following example:

```
DEFINE STREAM CCAJRNL WITH SCOPE=SYSTEM PARALLEL=(JRNL1,JRNL2) MINAVAIL=2
```

```

DEFINE DATASET JRNL1  WITH SCOPE=SYSTEM DSN=CCAJRNL.JRNL1 OLD
DEFINE STREAM  JRNL2  WITH SCOPE=SYSTEM GDG=J2 CONTROL=J2CTL
DEFINE DATASET J2      WITH SCOPE=SYSTEM DSN=CCAJRNL.GDGBASE.JRNL2 CATALOG -
                        GEN=+1 CYL PRI 500
DEFINE DATASET J2CTL   WITH SCOPE=SYSTEM DSN=CCAJRNL.GDGBASE.JRNL2.CTL OLD

```

Since data set JRNL1 has no target data set for a switch, when the JRNL1 data set is marked full, the number of available parallel stream members (MINAVAIL) drops to one, since only JRNL2 now has space available. Since the number of available members is now less than MINAVAIL the Online would stop with a CCAJRNL full message.

If before JRNL1 fills, a SWITCH STREAM CCAJRNL command were issued, the following messages would be produced:

```

M204.2712: MEMBER JRNL1 IS INELIGIBLE FOR SWITCHING
M204.2712: STREAM JRNL2 IS BEING SWITCHED
M204.2712: STREAM CCAJRNL - NOT ALL MEMBERS SWITCHED

```

Member JRNL1 was not switched. However, if JRNL1 has not filled, the parallel stream will remain open because the number of available members (MINAVAIL) is still 2. Stream JRNL2 always has a target data set to switch to since it is a GDG. Nevertheless, whenever JRNL1 fills, the minimum available members will be less than MINAVAIL and the stream will be closed and the run will terminate.

Because stream members are not locked before switch processing, and because all members are not switched at exactly the same time, the number of records in individual data sets of a parallel stream may not be identical. This is of no consequence to recovery or REGENERATE, but should be noted. However, the total number of records in each member is identical.

Previously, you had to concatenate all journals into one data set or specify a concatenated CCAGEN DD statement.

SWITCH STREAM command for concatenated streams

The same limitation exists for the last member in a concatenated stream. Since there is no additional member to switch to, the following message is issued:

```
M204.2712: STREAM streamname IS INELIGIBLE FOR SWITCHING
```

If all members of a stream are switched, the following message is issued:

```
M204.2712: STREAM CCAJRNL SWITCHED VIA COMMAND
```

Handling streams without records

You cannot switch a stream member that contains no records. So in the previous case, if the first G1 stream member has just become full and the newly

opened (second) GDG member contains zero records, then a SWITCH STREAM CCAJRNL command is not processed.

Since J1 is ineligible, and G1 (second member) is empty, no switch occurs. The messages issued are:

```
M204.2712: MEMBER J1 IS INELIGIBLE FOR SWITCHING
M204.2712: MEMBER G1 IS EMPTY AND CANNOT BE SWITCHED
M204.2712: SWITCH WAS UNSUCCESSFUL
```

Recovery parameters

Use the following recovery parameters when generating a journal:

Parameter	Set	Specifies...	Comments
FRCVOPT	During file creation or with the RESET command	File recovery options	If the file is included in system or media recovery, do not use FRCVOPT=X'04', which suppresses roll forward logging.
RCVOPT	On the JCLEXEC parameter or on User 0's parameter line	Type of information written to the journal	X'08' writes the roll forward information that is required for system and media recovery to CCAJRNL. X'09' also causes preimages to be written to the CHKPOINT data set for roll back recovery. Use this setting if you want to enable full recovery.

Writing error messages to the journal data set

You can control which error messages or classes of error messages to store in CCAJRNL, CCAAUDIT, or CCAJLOG using MSGCTL command parameters as described in the *Model 204 Parameter and Command Reference*, "MSGCTL: Setting message output".

ERRMSG - Setting the length of saved error messages

The ERRMSG parameter provides the ability to set the number of bytes to use for saved error messages—messages returned by \$ERRMSG and \$FSTERR. You can set ERRMSG to any value from 80 to 256—that length includes a count byte. The value is rounded up to an 8-byte multiple. For example, if you set ERRMSG=99, it will be rounded to 104, that value is reduced by 1 for the count byte, thus allowing up to 103 characters of an error message to be saved. See the *Rocket Model 204 Parameter and Command Reference* for more on the ERRMSG parameter.

Server size requirements for saved error messages

Increasing ERRMSG_L increases the requirement of the fixed table size of a server. This may necessitate an increase in your SERVSIZE settings. The size requirement for ERRMSG_L is:

$$3 * (ERRMSG_L - 80)$$

For example, increasing ERRMSG_L to its maximum of 256 would increase the fixed server requirement by $(3 * (256 - 80) - 1)$ or 527 bytes.

For a full update to the SERVSIZE formula, see “Calculating fixed table size” on page 47.

Using the CCAJRNL data set

During a production Online run, roll forward recovery information is written to the CCAJRNL data set. Additionally, a substantial amount of unformatted audit trail data is also written. When recovery is required and after roll back recovery has completed, that data set (during recovery its ddname is CCARF) is read and the roll forward information is extracted and used to reapply all committed updates up to the time of the system failure.

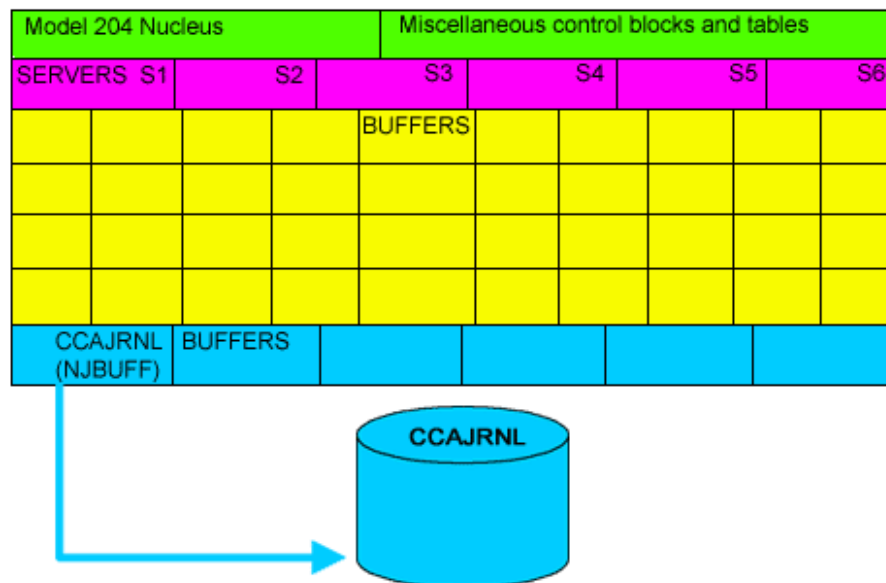


Figure 13-1. CCAJRNL set up to collect all roll forward and audit information.

Performance efficiencies using CCAJRNL

Choosing whether to define CCAAUDIT

Model 204 operates more efficiently using CCAJRNL without CCAAUDIT, because it is not also generating a separate audit trail. If you need printed

information and your site does not maintain CCAAUDIT, you can extract the audit trail from CCAJRNL in a separate step using AUDIT204.

If you need printed run information immediately upon completion of the job and you also need Roll Forward or Accounting facilities, create both the journal (CCAJRNL) and the audit trail (CCAAUDIT).

You can use CCAJRNL in place of CCAAUDIT when you do not need audit trail data immediately following run termination.

Allocating multiple journal buffers

Setting NJBUFF to a value ($\text{NSERVS} + \text{NSUBTKS} + 1$) allocates multiple journal buffers and ensures that a free buffer is always available for the journal. In z/VM, the NJBUFF parameter is stacked in the EXEC that issues all the FILEDEFs.

Using the CCAJLOG data set

If you allocate CCAJLOG, the unformatted audit trail data is written to it and only roll forward recovery data is written to CCAJRNL. Reducing the number of records written to CCAJRNL improves recovery performance and also reduces the likelihood of filling CCAJRNL, which would result in run termination.

- The number of buffers allocated for CCAJLOG is determined by the parameter, NLBUFF, which defaults to five. Only full blocks/buffers are written to CCAJLOG.
- The size of these buffers is determined by the BLKSIZE parameter defined for the CCAJLOG data set which may be as large as 32K.

You can produce a formatted CCAAUDIT report from CCAJLOG using the AUDIT204 utility. In this case, the CCAJLOG is referenced via the ddname CCAJRNL.

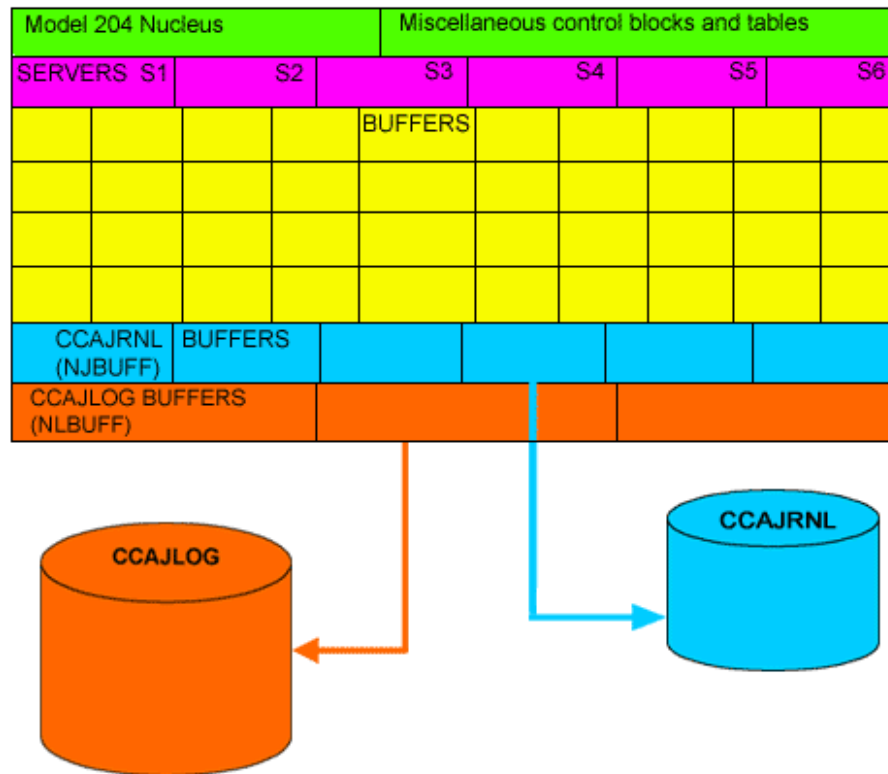


Figure 13-2. CCAJRNL collecting roll forward data; CCAJLOG collecting audit trail data

Defining the CCAJLOG stream

You can define a CCAJLOG using any of the stream definitions: concatenate, parallel, ring, or GDG; see the *Rocket Model 204 Parameter and Command Reference*, "DEFINE STREAM: Describing sequential I/O streams".

M204JLOG assembler exit

The assembler exit, M204JLOG, can be invoked if linked in. This exit is initiated when a SWITCH command is issued against a CCAJLOG GDG stream. The exit may be any AMODE and need not be reentrant.

Coding considerations

On entry, the registers contain:

R4 = A(GDG LIOD)

R9 = A(new switch control record) (see CRD dsect)

R13 = A(OSW save area)

R14 = return address

R15 = base address

Model 204 expects no output from the exit. All registers must be restored before return. If any Model 204 data structures are modified by the exit, unpredictable results may occur.

Note: If the user exit abends, Model 204 issues an error and produces a snap. This leaves Model 204 waiting for the switch to complete. At that point the switch will never complete, no further Online activity is possible, and the Online must be cancelled.

In addition, while the M204JLOG user exit is running, Model 204 cannot continue normal processing until a return from the exit is accomplished. For this reason, WAITS and I/Os inside the exit are strongly discouraged.

Sample M204JLOG Assembler exit

```
M204JLOG  CSECT
M204JLOG  AMODE 31
M204JLOG  TITLE 'TEST THE MODEL 204 CCAJLOG USER EXIT'
X10      EQU      10
X11      EQU      11
X12      EQU      12
X13      EQU      13
X14      EQU      14
X15      EQU      15
          STM      X14,X12,12(X13)      SAVE CALLERS REGISTERS
          LR       X12,X15              ESTABLISH BASE REGISTER
          USING    M204JLOG,X12
          LA       X10,SAVEAREA          GET A(LOCAL REGISTER SAVEAREA)
          ST       X10,8(,X13)           CHAIN OUR SAVEAREA TO CALLERS
          ST       X13,SAVEAREA+4        CHAIN CALLERS SAVEAREA TO OURS
          LA       X13,SAVEAREA          SET A(OUR SAVEAREA)
          WTO      'M204JLOG EXIT INVOKED, DOING SOMETHING'
* *****
* * CUSTOMERS MAY PLACE CODE HERE TO DO WHATEVER THEY DESIRE. * *
* *****
          WTO      'M204JLOG EXIT ENDING'
          L        X13,4(,X13)           RESTORE CALLERS SAVEAREA ADDRESS
          ST       X10,16(X13)           SET RETURN CODE (R15)
          LM       X14,X12,12(X13)      RESTORE CALLERS REGISTERS
          BR       X14                  RETURN TO CALLER
          DS       0D
SAVEAREA  DS      18F                  REGISTER SAVE AREA
          LTORG
          END
```

Model 204 roll forward recovery with CCAJLOG

Model 204 operates more efficiently during roll forward recovery, if you have defined CCAJLOG. This is because CCAJRNL now contains only roll forward recovery data; CCAJLOG contains unformatted CCAAUDIT data, which does not have to be read and ignored during the roll forward recovery since it has been removed from CCAJRNL.

Separating transaction (CCAJRNL) and auditing (CCAJLOG) information

Splitting audit trail records—messages and statistics—out from CCAJRNL and writing them to CCAJLOG improves recovery performance by reducing the size of CCAJRNL. This also reduces the likelihood of filling CCAJRNL. However, it does shift the possibility to CCAJLOG and if either CCAJRNL or CCAJLOG fills, the run comes down.

Regenerating files using CCAJRNL and CCAJLOG

Sending audit trail data to CCAJLOG also speeds up the REGENERATE processing, if that is required.

Considerations for CCAJLOG

By specifying CCAJLOG in the JCL or dynamically at the start of CCAIN, all messages and statistics are written to CCAJLOG. All recovery-type records are written to CCAJRNL.

- If you do not specify CCAJLOG, a CCAJRNL is maintained that collects recovery records, messages, and statistics.
- If CCAJLOG is specified but the open fails, the run terminates initialization.
- If CCAJLOG fills during a run, the run is terminated.

Set the NLBUFF parameter to specify the number of buffers to use for CCAJLOG. If you do not specify a value for NLBUFF, a default value of five is allocated. For more details, see the *Rocket Model 204 Parameter and Command Reference*, "NLBUFF: Number of buffers for CCAJLOG".

Using the journals correctly

If you create both a CCAJRNL file and a CCAJLOG file, you must use the CCAJLOG file as input to AUDIT204. And, you must continue to use the CCAJRNL file as input to REGEN and RESTART commands; otherwise, Model 204 issues one of the following messages:

```
M204:2515: CCAJRNL DATASET IS INVALID FOR AUDIT 204
```

```
M204:2515: CCAJLOG DATASET IS INVALID FOR {REGEN |  
RESTART}
```

If you create both a CCAJLOG and CCAJRNL and you use the CCAJRNL as input to AUDIT204, AUDIT204 issues the following message and stops processing:

```
NON-MESSAGE DATASET IS INVALID FOR CCAJRNL
```

Although UTILJ can handle a CCAJLOG data set, the utility will, of course, find only message and statistics type records in CCAJLOG. So, if you ask for recovery journal entries, type 1-6, the output is empty, since they do not exist on the CCAJLOG.

Because audit trail information is stored in CCAJLOG, if you have written a custom application for statistics reports, use CCAJLOG for input.

CCAJRNL data set record layout

The journal data set is composed of variable length records. Each record consists of a header, one or more journal entries, and a trailer. The format of the journal data set is subject to change with each release of Model 204.

Journal entry format

Journal entries contain audit trail, statistical, and control information, or information used for recovery. Statistics are recorded in the journal as fullword hexadecimal counters. Space is allocated in a given journal statistics entry for all possible statistics, even if they are not being kept. The values for statistics that are not maintained, or do not have relevance to the current configuration of Model 204, are represented as hexadecimal zeros.

Each entry has the same general format:

Bytes	Contents
0–1	Length
2	Entry type
3– <i>n</i>	Data

Offset 0 is a 2-byte hexadecimal field containing the length of the entry. (The only exception to this rule is the header entry, where the length is the length of the entire journal block.)

Offset 2 is a 1-byte field, which identifies the type of the entry: file, system, or user.

Offset 3 is a variable-length data portion. The size and layout depends upon the type of entry.

Table 13-2 summarizes the journal entry types, including formatting layouts and statistical information. Use the layout and statistics tables in Appendix A: “Using

System Statistics” for developing customized software to extract particular information from the journal data set.

Table 13-2. Summary of journal entry types

Type	Journal record
X'00'	Header/Trailer
X'01'	Recovery entry
X'02'	Recovery entry
X'03'	Recovery entry
X'04'	Recovery entry
X'05'	Recovery entry
X'06'	Recovery entry
X'07'	Unused
X'08'	System statistics
X'09'	User statistics
X'0A'	File statistics
X'0B'	Discontinued audit trail text
X'0C'	Initialization entry
X'0D'	Possibly continued audit trail text
X'0E'	Timestamp
X'0F'	Merged journal brackets

Number of lines in the journal data set

The number of lines in the journal are controlled by the CAUDIT, LAUDIT, and SYSOPT parameters:

CAUDIT parameter

The CAUDIT parameter controls physical input lines from a terminal, procedure, some IFAM arguments, or full-screen input. CAUDIT operates at physical line (card) level. It controls the auditing of input lines after line editing has been performed, but before line continuation is interpreted.

CAUDIT is set on any user parameter line at the beginning of CCAIN, and can be reset by a user with system manager privileges who logs in with that user number. (CAUDIT is generally used only by Technical Support staff in special instances.)

The value of CAUDIT is expressed as a sum of the options shown in the *Model 204 Parameter and Command Reference*, "CAUDIT: Input line auditing type".

LAUDIT parameter

The LAUDIT parameter controls logical input lines from a terminal or procedure, some IFAM arguments, and full screen information. LAUDIT is useful in reconstructing events that lead to a system crash.

Set LAUDIT on any user parameter (IODEV) line or at the beginning of CCAIN. For example:

```
PAGESZ=6184 , NUSERS=1 , NSERVS=1 , NFILES=20 ,  
LENQTB=15 , LAUDIT=5 , . . .
```

LAUDIT can be reset by a user with system manager privileges who logs in with that user number. See the *Rocket Model 204 Parameter and Command Reference*, "LAUDIT: Logical input line audit".

SYSOPT parameter

The SYSOPT=32 option controls output of information that relates to system initialization or to an IFAM function call. See the *Rocket Model 204 Parameter and Command Reference*, "SYSOPT: System options".

Sizing the journal buffer

In z/OS and z/VM only, you can specify the NJBUFF parameter on the EXEC statement to make journal entries of one user without interrupting other current users. Valid settings are the default NJBUFF=1 or, to ensure the availability of a free buffer:

$$\text{NJBUFF} = \text{NSERVS} + \text{NSUBTKS} + 1$$

In z/OS only, if the setting of the NJBUFF parameter is greater than 1, the setting is automatically recalculated using the previous formula and the initial value of NJBUFF is used to define the number of concurrent I/O operations (NCP) for the journal data set. Sizing NJBUFF to the value calculated in the formula guarantees the optimal number of Network Control Programs (NCP) for the journal data set.

If you use the NJBUFF option, specify BLKSIZE=6749 as the minimum journal buffer size to allow a full page to be written to a single journal buffer. The contents of a buffer are automatically written to CCAJRNL before the buffer is completely full.

Messages giving the number of journal buffers and the number of bytes allocated to the buffers are displayed at the end of the run.

Managing space requirements for all operating systems

The space requirement for CCAJRNL is reduced, if CCAJLOG is enabled. However, CCAJRNL must still be sized to prevent B37, D37, or E37 abends, which will terminate the run and require recovery to get files to a state of logical and physical consistency. The same applies to CCAJLOG: If it fills, the run terminates and recovery is required. Generous secondary extents allocated to these data set also help avoid overflow problems.

Additional managing space requirement on z/OS

At a z/OS site, the most comprehensive way to avoid data set full problems is to define both CCAJRNL and CCAJOG as generation data groups (GDGs). See “Perpetual journaling for z/OS” on page 419.

Introducing Model 204 statistics

A statistic is written to CCAAUDIT only if its value is nonzero. However, all statistics, even those with zero values, are written to CCAJRNL (or CCAJLOG).

Audit trail and journal statistics lines

The statistics generated on audit trail lines are valuable in a study of the performance of an individual user or of the Model 204 system as a whole. The effect of configuration changes within Model 204 and with other jobs in the operating system can be determined by comparing statistics gathered in Model 204 runs.

- Statistics described as percentages are multiplied by 10 (a value printed as 1000 represents 100 percent).
- Statistics described as averages are multiplied by 1000 (1000 represents 1.000).

Output lines

Statistics for each user and file active during a particular run are written to the journal and audit trail files in output lines that begin with ST \$\$\$\$. The output lines are written in groups of related lines that include:

- User lines, which you can monitor to identify active users and to help determine if individual applications are efficiently implemented
- File lines
- System lines, which you can watch—together with file lines—to signal shifting loads on the CPU, disks, and terminals

The type of line written—user, file, or system—is specified by the value of the first parameter after the journal header.

Identifying subtypes

The subtype—final, partial, performance, or since-last—within each type is specified by the value of the second parameter after the journal header. In the audit trail, the subtype is indicated by `USERID=user-id` (user statistics), `FILE=filename` (file statistics), and `SYSTEM=Model-204-version` (system statistics).

Within each major type of statistics line, subtypes are produced under specified conditions: some are always produced, others are optional. Optional subtypes are produced if the appropriate parameters are set during system initialization. See “Setting the NSUBTKS parameter” on page 302.

Monitoring statistics

The Model 204 statistics that you can monitor are introduced in Appendix A: “Using System Statistics”. The statistics are presented in alphabetical order with their stated purpose, followed by tables that identify their position in various layouts.

You can monitor statistics using one or more of the following methods:

- Print out the audit trail.
- View formatted or unformatted displays of all nonzero cumulative statistics for active users—individual, specified groups, and all users—and system activities.
- Use the AUDIT204 or UTILJ utility program to extract and summarize statistics lines from the journal data set. See “Introducing the AUDIT204 utility” on page 306 and “Using the UTILJ utility” on page 393.
- Write optional records to the System Management Facilities (SMF) data set; see “System Management Facilities” on page 591.

Setting the NSUBTKS parameter

Gathering partial or performance statistics lines requires a Model 204 pseudo subtask. Model 204 uses pseudo subtasks to perform actions that must be done regularly but that cannot be assigned to a specific user. To accommodate partial and performance statistics lines you might need to increase the NSUBTKS parameter on User 0’s parameter line, which controls the maximum number of pseudo subtasks that can be allocated. Pseudo subtasks are described in “Pseudo subtasks” on page 132.

Setting parameters to collect certain statistics

You must set various parameters to collect certain statistics. Table 13-3 lists the statistics and the corresponding parameter(s).

Table 13-3. Parameters to set to collect certain statistics

Statistic	Parameter to set to nonzero
BLKCFRE	CFRJRNL and CFRLOOK
BLKI	RPTCNT and SMPLTIM
BLKO	RPTCNT and SMPLTIM
BLKRLK	CFRJRNL and CFRLOOK
DKRR	LRUPG
DKSDIR	LDKBMW
DKSDIRT	LDKBMW
DKSWRP	LDKBMW
DKSWRPT	LDKBMW
LONGUPDTIME(MS)	MAXUD
LONGUPDTS	MAXUD
PNDGTIME	DKUPDTWT
REDY	RPTCNT and SMPLTIM
RUNG	RPTCNT and SMPLTIM
SMPLS	RPTCNT and SMPLTIM
SWPG	RPTCNT and SMPLTIM
USRS	RPTCNT and SMPLTIM
WTCFR	CFRJRNL and CFRLOOK
WTRLK	CFRJRNL and CFRLOOK
WTSV	RPTCNT and SMPLTIM

Audit trail format

Whether printed to your terminal from CCAAUDIT or CCAJRNL (or CCAJLOG) is processed by AUDIT204, the format of an audit trail line is:

Syntax `yydddhmmss nnn sss xxxxx tt (...) audit-trail-text`

Where

- `yyddd` is the Julian year and day.
- `hhmmss` is the time of day in hours, minutes, and seconds.

- *nnn* is a counter to distinguish lines produced in the same second.
- *sss* is the number of the server currently handling the user. (Leading zeros are suppressed.)
- *xxxxx* is the 5-digit number of the user associated with the audit trail line. (Leading zeros are suppressed.)
- *tt* is code for the type of audit trail line. The *tt* codes are shown in Table 13-4

Example

```
05263092241 10 0 1 AD...M204.0763: BEGIN FILE INITIALIZA-
TION:
```

Table 13-4 lists the code and purpose of the audit trail lines.

Table 13-4. Types of audit trail lines

Code	Displays...
AD	Special information about the run, job step return code, status of Model 204 files, user or the password table, the SNA Communications Server (formerly VTAM) Interface, or messages sent from an HLI program via the IFERR call. Note that the high-water mark of CCATEMP usage (SCMAX) also reports the total number of pages allocated to CCATEMP.
CI	Physical input line from a user's terminal or some IFAM arguments.
CP	Physical input line from a procedure.
CS	Physical line of full-screen input (see LS below).
ER	Error message sent to the user.
LI	Logical input line from a user's terminal or some HLI arguments.
LP	Logical input line from a procedure.
LR	Read images from a terminal.
LS	Full-screen information. For a screen, each line lists one input value. For a menu, the line provides the menu selection number.
MS	Informational message sent to the user (not an error).
OI	Input line from the operator's console.
OO	Message sent to the operator's console.

Table 13-4. Types of audit trail lines (Continued)

Code	Displays...
RK	Special information that relates to system initialization or to the record of a call to the following types of function: <ul style="list-style-type: none"> • Connect★ • HLI • MQ/204 • PQO • SQL • UL/DB2
QT	Lines of SQL statement level processing
ST	Utilization statistics (see Appendix A).
US	Line of special information directed to the audit trail by means of the User Language AUDIT statement.
XX	Continuation of the previous line.

Generating an audit trail

Overview of audit trail parameters

An audit trail is generated if:

- The SYSOPT parameter includes the 128 specification
- CCAAUDIT DD statement

The LAUDPROC parameter controls the length of procedure names that appear in since-last statistic lines of the audit trail. You can reset LAUDPROC to a low or high value. A low value conserves memory. A high value captures long procedure names.

z/VSE and the audit trail

In a z/VSE environment, activate the audit trail by incorporating the following specifications into the JCL:

- You must set UPSI to 1xxxxxxx (SYSOPT=128).
- You must write the audit trail file to either a disk device or a print device:
 - Assign a disk device using a DLBL and EXTENT statement for file name CCAUDIT (retention 0 is recommended).
 - Assign a print device using symbolic unit SYS008. (You cannot assign SYS008 to the same physical device as SYSLST.)

If the DLBL and EXTENT for CCAUDIT are provided in the JCL, the audit trail is written to disk regardless of the assignment of SYS008.

The following example shows a job stream for an ONLINE configuration with the audit trail on disk:

```
// JOB ONLINE
...
// DLBL CCAUDIT,'audit trail file-id',0
// EXTENT SYSnnn,balance of extent information
// ASSGN SYSnnn,X'cuu'
...
// UPSI 10000000
// EXEC ONLINE, SIZE=AUTO
...
/ &
```

After the audit trail has been written to disk, you can print it using the UTLA utility program supplied with the Model 204 DBMS. The UTLA utility is discussed in the *Model 204 z/VSE Installation Guide*.

Use the following JCL:

```
// JOB UTLA PRINT MODEL 204 AUDIT TRAIL
// DLBL M204LIB,'M204.PROD.LIBRARY'
// EXTENT SYSnnn,...
// LIBDEF PHASE.SEARCH=M204LIB.V411
// DLBL CCAUDIT,'audit trail file-id'
// EXTENT SYSnnn,balance of extent information
// ASSGN SYSnnn,X'cuu'
// ASSGN SYS005,SYSLST
// EXEC UTLA,SIZE=AUTO
/ &
```

z/VM and the audit trail

In the z/VM/CMS environment, define CCAUDIT as one of the following:

- CMS file:

```
FILEDEF CCAUDIT DISK ONLN CCAUDIT A
```

- Service machine virtual printer:

```
FILEDEF CCAUDIT PRINTER
```

Introducing the AUDIT204 utility

The AUDIT204 utility program produces the following:

- Complete or partial audit trail from a journal—CCAJRNL or CCAJLOG, as ddname CCAJRNL—produced by a Model 204 run
- Report based on statistics that appear on user logout and partial lines or file closed and partial lines

- Statistical analysis of the evaluation of User Language requests

The format of AUDIT204 reports is determined by AUDIT204 commands and parameters you enter in:

- z/OS CCAIN data stream
- z/VM (CMS) AUDIT204 CCAIN file
- DOS SYSIPT statement

The basic AUDIT204 input commands are:

AUDIT204 command	For...
ANALYZE	Statistical analysis of the evaluation of User Language requests
FORMAT	Printed audit trail
REPORT	Breakdown of user and file statistics lines containing optional price calculations

Input to AUDIT204

Input to AUDIT204 is free form, located in columns 1–71. The following rules apply:

- Use blanks to separate terms.
- Type each parameter on a separate line.
- You can type parameters for a single command in any order.
- You can add subparameters for some parameters (PRICE and RENAME) by naming the subparameters in the form of character strings:
 - If a name contains a blank or a single quotation mark, you must enclose the name in single quotation marks.
 - Use double quotation marks within the name.
 - Use blanks around the equal signs in the PRICE and RENAME parameters.
 - Use blanks to separate subparameters.
 - If more than one line is required for the subparameters, place a non-blank character in column 72 or repeat the parameter on a new line.
- Any characters appearing in columns 73–80 are ignored.
- You can repeat any parameter.
- Precede each comment by an asterisk (*) in column 1.
- The effect of parameters containing lists of subparameters is cumulative.

- The last value of parameters without subparameter lists is the value used.

ANALYZE command

The ANALYZE command obtains a statistical analysis report on the evaluation of User Language requests by AUDIT204. The complete report comprises an analysis for each individual account and the system as a whole.

Since-last evaluation lines calculate the mean and standard deviation for each since-last statistic:

Measure	Represents...
Mean	Center of a set of data points (usually referred to as the average value of the set), obtained by summing the values and dividing by the number of values.
Standard deviation	How spread out the data is around the center. In particular, it is useful to know whether the data is tightly clustered around the mean, or more dispersed. A small standard deviation indicates that any given value is likely to be close to the mean, and a large standard deviation implies that a value is more likely to be far away from the mean.

You can obtain more information about the mean and the standard deviation from any standard statistics textbook.

Syntax Use the following syntax for the ANALYZE command:

```
ANALYZE [USERID | NOUSERID]
        [USERIDS userid1...useridn]

or

ANALYZE [ACCOUNT | NOACCOUNT]
        [ACCOUNTS account1...accountn]

        [parameters]
```

Where Use the following parameters with the ANALYZE command:

Parameter	Specifies...
ACCOUNT or NOACCOUNT	Whether or not to do an individual account analysis. The ACCT or NOACCT parameters are equivalent to ACCOUNT or NOACCOUNT, respectively.

Parameter	Specifies...
ACCOUNTS	Login account name(s) to list in the analysis report. Reports are produced only for the specified accounts and only the since-last evaluation lines for those accounts that are included in the systemwide report. Specifying ACCOUNTS sets the ACCOUNT parameter when neither ACCOUNT nor NOACCOUNT was specified.
OMIT name1 name2...	Since-last statistic(s) to list, but omit from the analysis report. The report contains all since-last statistics that are not explicitly omitted.
RENAME name=newname	New names to since-last statistics as column headings for an analysis report.
TIME [yydddhmmss / yydddhmmss / cyydddhmmss / cyydddhmmss]	Time range from which since-last evaluation lines are taken.
SYS (the default) or NOSYS	Whether or not to do a systemwide analysis using since-last evaluation lines of accounts for which an individual analysis is done.
USERID (the default) or NOUSERID	Performing or suppressing an individual user ID analysis. Totals for a systemwide analysis are still accumulated if NOUSERID is specified.
USERIDS	User(s) for which analysis is requested. The USERIDS keyword is required in addition to the USERID keyword. If neither USERID nor NOUSERID has been specified, specifying USERIDS sets the USERID parameter. Reports are produced only for the specified users and only the since-last evaluation lines for those users who are included in the systemwide report. AUDIT204 uses the values specified in the USERID parameter as a pattern and finds ALL values that match. For example, USERIDS NANCY KATHY is equivalent to REPORT USERID USERIDS NANCY* KATHY*.

Usage note

The ANALYZE command for AUDIT204 can process only a single journal, CCAJRNL. You cannot use the ANALYZE command against the CCAJLOG file.

FORMAT command

FORMAT prints an audit trail from the journal.

Syntax

```
FORMAT TIME {yydddhmmss/yydddhmmss |
             cyydddhmmss/cyydddhmmss}
```

```
TYPE type1
USER usernum
USERID userid
YEARFORM {2 | 4}
```

Where

Use the following parameters with the FORMAT command:

Parameter	Specifies...
TIME	Time range to print. The format of the time specification corresponds to the time stamp printed with each audit trail line. You can specify start and end times in either order. If either time is out of the range covered by the journal, it is corrected to match the actual start or end of the journal. If both specified times are before and after the period covered by the journal, nothing is printed. <ul style="list-style-type: none"> <i>yyddd or cyyddd</i> is the year and Julian date. <i>hhmmss</i> is time based on a 24-hour clock.
TYPE	Type of lines to print, such as AD or MS (see Table 13-4 on page 304).
USER	User for whom the printed audit trail lines are generated. Specify user numbers with or without leading zeros.
USERID	One- to ten-character name that identifies the Model 204 user about whom the printed audit trail lines are generated.
YEARFORM	Two- or four-digit year output. The default is 2.

Example

The options specified in the following example restrict the printing of the audit trail to MS (message) lines for USER 01 created on September 22, 1998 between 10:41 A.M. and 10:45 A.M.:

```
FORMAT TYPE MS, ER
TIME 98265104100/98265104500
USER 01
```

Usage

The FORMAT command can be used against a CCAJRNL file or a CCAJLOG file. Examine the code examples that begin in “Samples of a z/OS job stream” on page 314.

REPORT command

The REPORT command prints user and file statistics in a tabular format and computes costs by account or by file for billing purposes. AUDIT204 reports have a 2-line header, as shown in the following example, that reflect the circumstances of your site.

```
AUDIT204 UTILITY - VERSION 7.10a,
```


DATE/TIME OF RUN: 10/22/208 11:31:38

Syntax

```
REPORT USERID [parameters]
        ACCOUNT
        FILE
```

Where

The following subcommands specify the type of report to print:

Subcommand	Specifies printing a report from...
ACCOUNT	User statistics lines for specific accounts
FILE	File statistics
USERID	User statistics lines

The following parameters can be used with USERID, ACCOUNT, and FILE:

Parameter	Specifies...
IND (the default) or NOIND	Printing or not printing individual lines. Totals are still accumulated if NOIND is specified.
OMIT	Statistic(s) to omit from the report. You must enter the actual name of the statistic, not what appears in a column header in a report. See Table A-1 on page 548 for an alphabetical listing of statistics.
PARTIAL or NOPARTIAL	Generation of partial or complete statistics lines. Use: <ul style="list-style-type: none"> PARTIAL when a report is generated from a journal that did not terminate successfully. NOPARTIAL to avoid processing overhead, when a complete journal is available.
PRICE	Dollar charge up to five decimal places to any statistic. Decimal point, and leading and trailing zeros are optional. For example: PRICE CPU=0.05 DKRD=.03 DKWR=.035
RENAME	Column heading names up to nine characters without truncation. The new name completely replaces the old name and must be used if name is specified on other parameters.
TIME	Time range for statistics line reporting. One of the following formats is required: <ul style="list-style-type: none"> yydddhmmss/ yydddhmmss cyydddhmmss/cyydddhmmss
TOTAL (the default) or NOTOTAL	Totalling or suppression of totalling for individual statistics and costs.

Example Totals are printed after individual statistics lines if both IND and TOTAL are in effect. Each line must contain only one parameter, as in the following examples:

```
REPORT USERID NOIND
      PARTIAL
      TIME yydddhhmmss/yydddhhmmss
      RENAME name=newname
      NOTOTAL
      USERIDS account
```

```
REPORT FILE FILES filename
      OMIT statistic
      PRICE statistic=xx.xx
```

The subkeywords ACCOUNTS, FILES, and USERIDS can be used with REPORT subcommands ACCOUNT, FILE, and USERID as in the following examples:

```
REPORT USERID USERIDS JAMES STEVE SANDRA
      ACCOUNT ACCOUNTS JAMES STEVE SANDRA
      FILE FILES CLIENTS VEHICLES DAILY
```

Usage note: The REPORT command for AUDIT204 can process only a single journal: CCAJRNL. You cannot use against a CCAJOG file.

Using the AUDIT204 utility

CCAJRNL and CCAJLOG data sets as input to utilities

When you need a CCAAUDIT report, use the CCAJLOG data set as input to the AUDIT204 utility, under the ddname of CCAJRNL. You can also use CCAJLOG as input to the UTILJ utility, again under the ddname of CCAJRNL; however, the only useful report produced is a histogram of the number of blocks and block sizes that were written to the CCAJLOG data set.

CCAJRNL is the input data set provided to roll forward recovery (ddname=CCARF), media recovery (ddname= CCAGEN), and UTILJ (ddname=CCAJRNL).

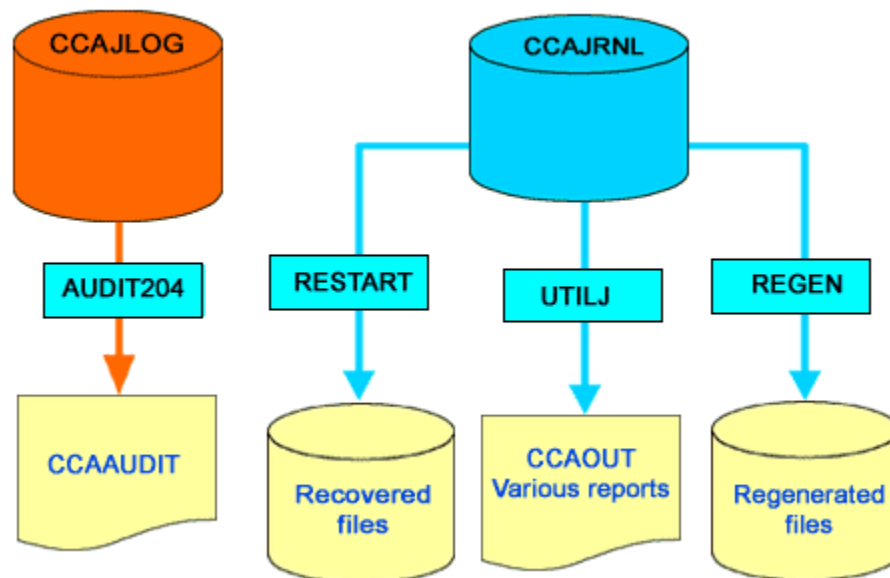


Figure 13-3. Utilities using CCAJLOG and CCAJRNL data sets as input

Journal stream configurations

AUDIT204 can process only basic journal streams (single data sets). Members of concatenated, parallel, ring, or GDG streams must be processed one at a time.

If there is a wraparound during the output processing of a ring stream, use the offload stream as input to AUDIT204. You can also use regular z/OS concatenation to print the contents of a concatenated stream.

z/OS JCL for AUDIT204

The following considerations apply to z/OS JCL for AUDIT204:

- Use a SORTLIB DD statement to identify the library in which the sort program is stored, if necessary.
- You might need sort work (SORTWKxx) data sets.
- SORTIN, SORTOUT, and SYSOUT statements are necessary only for statistical reports. A sort program must be available:
 - SORTIN contains all the user and file statistics entries extracted from the journal. The size of the data set depends on the number of statistics entries extracted. Each user statistics entry is 198 bytes long. Each file entry is 82 bytes long. Partial statistics entries are not copied to SORTIN.

- SORTOUT contains the sorted statistics entries from SORTIN. SORTOUT and SORTIN must be the same size.
 - SYSOUT is the SORT message data set.
- STATIN, STATOUT, and SYSOUT statements are necessary only for statistical analysis reports:
 - STATIN contains all the since-last evaluation statistics entries extracted from the journal. The size of the data set depends on the number of since-last evaluation statistics extracted. Each since-last evaluation statistics entry is 188 bytes long.
 - STATOUT contains the sorted statistics entries from STATIN. STATOUT and STATIN must be the same size.
- Statistics work (STATWKxx) data sets might be needed when using the ANALYZE command, depending on the amount of data generated.
- CCAJRNL describes the data set containing the journal produced by a Model 204 run.
- AUDIT204 does not support streams.
- CCAAUDIT contains the audit trail, statistics report, and statistical analysis report. If you request both the audit trail and statistics, the audit trail appears first.

The value of the BLKSIZE parameter in the CCAAUDIT DD statement is truncated at 4 more than a multiple of 137. A BLKSIZE value of 141 is the default and minimum value.

Any LRECL and RECFM specifications are ignored; these parameter values are forced to be 137 and VBA, respectively.

Samples of a z/OS job stream

The following sample job streams run AUDIT204. These examples differ in the use of the journal processed, CCAJRNL or CCAJLOG, and the presence or absence of AUDIT204 commands: ANALYZE, FORMAT, and REPORT.

Where only CCAJRNL is defined, all AUDIT204 commands are valid

```
//AUDIT      EXEC   PGM=AUDIT204
//STEPLIB    DD     DSN=M204.LOADLIB,DISP=SHR
//SYSUDUMP   DD     SYSOUT=A
//CCAJRNL    DD     DSN=M204.CCAJRNL,DISP=SHR
//CCAAUDIT   DD     SYSOUT=A
//SORTIN     DD     DSN=&&SORTIN,UNIT=SYSDA,SPACE=(TRK,nn),
//            DISP=(NEW,DELETE)
//SORTOUT    DD     DSN=* .SORTIN,VOL=REF=* .SORTIN,DISP=(OLD,PASS)
//STATIN     DD     DSN=&&STATIN,UNIT=SYSDA,SPACE=(TRK,nn),
//            DISP=(NEW,DELETE)
```

```
//STATOUT DD DSN=* .STATIN, VOL=REF=* .STATIN, DISP= (OLD, PASS)
//SYSOUT DD SYSOUT=A
//CCAIN DD *
FORMAT
ANALYZE
REPORT ACCOUNT
.
. ---- user options
.
/*
```

Where CCAJLOG is defined and you want to extract CCAAUDIT

```
//AUDIT EXEC PGM=AUDIT204
//STEPLIB DD DSN=M204.LOADLIB, DISP=SHR
//SYSUDUMP DD SYSOUT=A
//CCAJRNL DD DSN=M204.CCAJLOG, DISP=SHR
//CCAAUDIT DD SYSOUT=A
//SORTIN DD DSN=&&SORTIN, UNIT=SYSDA, SPACE= (TRK, nn) ,
// DISP= (NEW, DELETE)
//SORTOUT DD DSN=* .SORTIN, VOL=REF=* .SORTIN, DISP= (OLD, PASS)
//STATIN DD DSN=&&STATIN, UNIT=SYSDA, SPACE= (TRK, nn) ,
// DISP= (NEW, DELETE)
//STATOUT DD DSN=* .STATIN, VOL=REF=* .STATIN, DISP= (OLD, PASS)
//SYSOUT DD SYSOUT=A
//CCAIN DD *
FORMAT
.
. ---- user options
.
/*
```

Where both CCAJRNL and CCAJLOG are defined and you want a statistical analysis

```
//AUDIT EXEC PGM=AUDIT204
//STEPLIB DD DSN=M204.LOADLIB, DISP=SHR
//SYSUDUMP DD SYSOUT=A
//CCAJRNL DD DSN=M204.CCAJRNL, DISP=SHR
//CCAAUDIT DD SYSOUT=A
//SORTIN DD DSN=&&SORTIN, UNIT=SYSDA, SPACE= (TRK, nn) ,
// DISP= (NEW, DELETE)
//SORTOUT DD DSN=* .SORTIN, VOL=REF=* .SORTIN, DISP= (OLD, PASS)
//STATIN DD DSN=&&STATIN, UNIT=SYSDA, SPACE= (TRK, nn) ,
// DISP= (NEW, DELETE)
//STATOUT DD DSN=* .STATIN, VOL=REF=* .STATIN, DISP= (OLD, PASS)
//SYSOUT DD SYSOUT=A
//CCAIN DD *
ANALYZE
```

```
REPORT
.
.  ---- user options
.
/*
```

z/VSE JCL for AUDIT204

The following considerations apply to z/VSE:

- Under z/VSE, output is sent to SYSLST.
- CCAJRNL can be on tape or disk. If CCAJRNL is on tape, SYS004 must be assigned to a tape drive.
- Running AUDIT204 might require two sorts:
 - REPORT command requires a SORTIN and a SORTOUT data set.
 - ANALYZE command requires a STATIN and a STATOUT data set.
- SORTIN, SORTOUT, STATIN, and STATOUT are work files and must be on disk.
- SORTWK1 data set is required.
- Statistics work (STATWK1) data set might be needed when using the ANALYZE command, depending on the amount of data generated.

Sample z/VSE job stream

The following sample job stream runs AUDIT204:

```
// JOB AUDIT204
// DLBL M204LIB,'M204.PROD.LIBRARY'
// EXTENT SYSnnn,...
// LIBDEF PHASE.SEARCH=M204LIB.V210
// DLBL CCAJRNL,'CCAJRNL' Note 1
// EXTENT SYSnnn,...
// DLBL SORTIN,'AUDIT204.SORTIN',0 Note 2
// EXTENT SYSnnn,...
// DLBL SORTOUT,'AUDIT204.SORTOUT',0 Note 2
// EXTENT SYSnnn,...
// DLBL STATIN,'AUDIT204.STATIN',0 Note 3
// EXTENT SYSnnn,...
// DLBL STATOUT,'AUDIT204.STATOUT',0 Note 3
// EXTENT SYSnnn,...
// DLBL SORTWK1,'SORTWK1',0 Note 4
// EXTENT SYSnnn,...

// EXEC AUDIT204,SIZE=(AUTO,xxK) Note 5
INCLUDE IFYLSQRT
```

```

* Insert AUDIT204 CCAIN commands here
FORMAT
REPORT ACCOUNT
REPORT FILE
REPORT ACCOUNT ACCOUNTS MAGGIE MARY DAVE
/*
/&

```

- Usage notes**
- If you use tape instead of the DLBL and EXTENT for CCAJRNL, use the following JCL:

```

// TLBL CCAJRNL
// ASSGN SYS004,X'cuu'

```

- DLBL and EXTENT are necessary if you use either the REPORT or the ANALYZE command.
- You must specify a SIZE parameter in the EXEC statement with AUTO and you need some additional storage for the sort program. To determine the amount of additional storage needed for the sort program, refer to IBM's *SORT/MERGE Programmer's Guide*.

(z/VM) CMS JCL for AUDIT204

The CMS AUDIT204 utility program prints a Model 204 journal file and produces a statistical report from information in a journal file:

- If a statistics report is produced, a SORT utility that can be invoked dynamically must be available for use.
- AUDIT204 output is produced either as a printer spool file or in a CMS disk file.

The following EXEC initiates AUDIT204:

Syntax `AUDIT204 [datasetname] [filename filetype] filemode`

Where The following arguments specify:

Argument	Specifies...
<i>datasetname</i>	Name of the journal data set on a variable-format disk, with the qualifiers separated by blanks. If no data set name is specified, it is presumed that the name of the journal data set is M204.JOURNAL.
<i>filename</i> and <i>filetype</i>	Name and type of the journal file on a CMS-format, or a variable format for z/OS and z/VSE, disk.

Argument	Specifies...
<i>filemode</i>	Mode of the disk holding the journal file to be processed. The input to AUDIT204 contains the commands selected from those described earlier and is included in the AUDIT204 CCAIN.

Searching multiple tapes for journal and off load data

When a journal or offload data set is on multiple tapes, you do not need to mount all of them for AUDIT204 processing. You can process a subset of the tapes, as long as you mount them in the order in which they were originally generated.

For example, suppose that your CCAJRNL data set consists of five tapes labeled TAPE01 through TAPE05. You are sure that the information you want to analyze begins on the third tape or later. In this case you can save processing time by specifying only TAPE03, TAPE04, and TAPE05 in the DD statement for the CCAJRNL data set.

14

Storing Diagnostic Information (CCASNAP and CCAMDMP)

In this chapter

- Overview
- SNAPCTL parameter
- Handling error messages with MSGCTL command
- Operating system requirements
- Using unformatted system dumps (z/OS)

Overview

The CCASNAP file stores the contents of SNAP dumps, which identify program or file integrity problems. In the case of a severe error, you can also use the SYSUDUMP file.

System dumps are normally indicated by the user abend code 2749. Dumps are also generated before snaps to preserve subtask information.

This chapter summarizes the SNAPCTL parameter and the MSGCTL command, which produce SNAPS. Requirements for producing dumps in z/OS, z/VSE, and z/VM systems are discussed.

The last section of this chapter explains how to create multiple SYSMDUMP data sets in z/OS environments.

Complete information about the SNAPCTL parameter and the MSGCTL command is given in the *Model 204 Parameter and Command Reference*.

SNAPCTL parameter

In z/OS and CMS environments, you can generate SNAP dumps using the SNAPCTL parameter in the User 0 input stream. SNAPCTL specifies the actions taken when Model 204 encounters an error that requests a SNAP.

A user with system manager privileges can set SNAPCTL on the User 0 parameter line or reset it with the RESET command. Settings allow a range of options from taking a complete SNAP to taking no SNAP and from producing no dump to a complete region dump. See the *Rocket Model 204 Parameter and Command Reference*.

Although SNAPCTL parameter settings correspond to settings available on the MSGCTL command, which are summarized in Table 14-1, SNAPCTL processing takes precedence.

Handling error messages with MSGCTL command

The MSGCTL command specifies the action performed by Model 204 for each Model 204 error message. MSGCTL options and the actions they perform are listed in Table 14-1.

Table 14-1. MSGCTL command options

Option	Performs this action...
AUDIT	Puts the specified message on the audit trail as an AD line.
COUNT	Increments, by one, the message count whenever this message is issued. If the message count exceeds the value of the ERMX parameter, the user's session and processing are stopped.
DUMPALL	Dumps the entire Model 204 region
NOACTION	Ignores the original option that was assigned the message and returns to main processing.
NOAUDIT	Suppresses the auditing of a specific error message.
NODUMP	Does not generate a dump
SNAP	Produces a SNAP
SNAPALL	Similar to SNAP
SNAPPD	Produces a SNAP that includes registers, module link map, allocated storage map, pushdown list trace, KOMM, disk buffers containing file pages held by the current user, and patch information
SNAPSEL	Produces a SNAP similar to that of SNAPPD along with additional specified output, such as table disk buffers

Complete information about the MSGCTL command is in the *Model 204 Parameter and Command Reference*.

Introducing message types

By design each Model 204 message is assigned a type—AD, ER, MS or RK—and assigned separately whether the message is counted or not. When the occurrence of a counted message equals the ERMX parameter value, processing stops and the user stopped.

The message types—AD, ER, MS or RK—are mutually exclusive. Usually, the ER messages are counted and the other message types are not. Each processing option is handled independently of other options, as each option is designed to manage a single function.

On a message by message basis, use the MSGCTL command to change message processing options to more precisely accommodate the needs of your site and applications.

Adjusting error messages at your site

Using the MSGCTL command options, you can manipulate the message type and whether the message is counted. Using the option:

- AUDITxx and NOAUDITxx options, you can manipulate the type of message and where it appears in the audit trail.
- COUNT or NOCOUNT option, you can manipulate whether the message counts toward the ERMX total.

Caution: Rocket Software strongly recommends the you do *not* apply the NOCOUNT option to messages that are part of User Language compilation processing. Counting errors typically terminate the compilation process. In instances where you make compilation errors NOCOUNT, compilation is not terminated. Failure to stop compilation can result in invalid information being made available to the evaluation process resulting in subsequent snaps.

Modifying message type

You can modify the assigned message type as shown in the following table.

Message type	Option applied	Reverts to...
AD	NOAUDITAD	MS
ER	NOAUDITER	RK
MS	NOAUDITMS	Not be audited (NOAUDIT)
RK	NOAUDITRK	AD

The NOAUDITxx option processes only a message of type xx. However, the NOAUDIT option can process any message type. Conversely, AUDITxx can process any message type and results in the message becoming type xx.

For example, consider the M204.1030 INVALID COMMAND message, which is designed as a counted, ER type message.

- If at your site you turn off all auditing and journaling for that message, you would issue the following command:

```
MSGCTL M204.1030 NOAUDIT
```

The message would no longer appear on the audit trail. However, it would still be a counted error and appear at the user terminal.

- Suppose at your site you issue the following command:

```
MSGCTL M204.1030 NOAUDIT NOCOUNT
```

The message would no longer appear on the audit trail and it would not be counted toward the maximum number of errors (ERMX). However, it would still appear at the user terminal.

Understanding MSGCTL command options

NOACTION option

The NOACTION option suppresses almost all error message processing. When an error is generated, Model 204 checks an indicator flag to determine if that error message needs to be processed and whether the processing options have been changed via MSGCTL processing. If an error is found to have no processing requirements—that is, NOACTION—processing returns immediately to the next instruction. NOACTION has no effect on the following messages:

- Counted messages that occur during compilation
- Initialization, start up, and restart
- Termination

The decreased number of instructions necessary to process the NOACTION option versus the NOAUDIT option results in a large CPU processing reduction.

Caution: If you allow User Language programs to continue processing past an error that has been suppressed with the NOACTION option of the MSGCTL command, you must expect abends, incorrect database results, recovery errors, the inability to recover or other unpredictable consequences. Rocket Software strongly recommends that you consider carefully the circumstances, before you introduce the NOACTION option.

AUDITxx options

These options can be used for any message type—AUDITER, AUDITRK, AUDITAD, and AUDITMS—to change the current message type to another xx type message and its related processing options.

The NOAUDIT option functions for a message of any type, suppressing the auditing of the designated message.

NOAUDITxx options

The NOAUDITxx options function only for a message of that designated xx type.

Option	Changes message type	To message type
NOAUDITER	ER	RK
NOAUDITRK	RK	AD
NOAUDITAD	AD	MS
NOAUDITMS	MS	NOAUDIT

Independent option assignment

For example, message M204.1030: INVALID COMMAND is designated to be processed as both a counting error and an ER type message.

If you wish to turn off auditing and journaling for this message, issue the following command:

```
MSGCTL M204.1030 NOAUDIT
```

The message will no longer be written to the journal or the audit trail. However, it remains a counting error and appears on the user's terminal.

To change this message so that it is also not a counting error, issue the following command:

```
MSGCTL M204.1030 NOAUDIT NOCOUNT
```

Again, the message will no longer be written to the journal or audit trail, and now it is not be counted toward the ERMX limit. However, it still appears on the user's terminal.

Using the SNAPFAIL and SNAPFLIM parameters

The snap formatter is capable of recovering first-level as well as second-level errors without disabling the Online. The errors in question are program exceptions in unexpected places. For example, when the snap formatter assumes a pointer is valid, but it is not. This avoids a 4095 ABEND, followed by a RECURSIVE ENTRY TO ESTAE, which would bring down the Online.

The parameters, SNAPFAIL and SNAPFLIM, identify snap failures that have happened and set a limit on the number of failed snaps. (See *Model 204 Parameter and Command Reference* for parameter details.)

At the completion of recovery from a failed snap, one of the following M204.1449 messages is sent to the operator and CCAJRNL or CCAJLOG, as well as saved in the VIEW ERRORS table.

- The following error message increases the value of SNAPFAIL by one:

```
M204.1449: ERROR WHILE PROCESSING CCASNAP
```

- The following error message indicated that snap formatting has been disabled, because the value of SNAPFAIL equals the value of SNAPFLIM.

```
M204.1449: ERROR WHILE PROCESSING CCASNAP, CCASNAPS  
DISABLED
```

Both versions of message 1449 sets the Online and Batch return codes to 100.

Operating system requirements

z/OS

In a z/OS environment, the CCASNAP and SYSUDUMP files are normally specified as SYSOUT type data sets:

```
//CCASNAP DD SYSOUT=A  
//SYSUDUMP DD SYSOUT=A
```

z/VSE

In a z/VSE environment, SNAP output prints on the logical unit SYSLST. Output appears on the output report in the order in which it is generated.

z/VM

You can dynamically create a VMDUMP of the Model 204 service virtual machine using the z/VM Control Program VMDUMP command, if you detect a serious Model 204 error. The dump file is sent to the virtual machine designated as the target for VMDUMPs, as defined in the SYSTEM CONFIG file.

A FILEDEF for CCASNAP is required to produce VMDUMP. VMDUMP files are generated depending on the setting of the Model 204 SNAPCTL option. You can specify the DUMMY operand in the file definition for CCASNAP, which is adequate to cause VMDUMP production. If you define CCASNAP as other than DUMMY, Model 204 generates formatted SNAP information that augments the VMDUMP.

You can define CCASNAP as one of the following:

- CMS file:

```
FILEDEF CCASNAP DISK ONLINE CCASNAP A
```

- Service machine virtual printer:

```
FILEDEF CCASNAP PRINTER
```

Some Model 204 problems require a manual VMDUMP of the service virtual machine. The following command is required when Model 204 requests a VMDUMP:

```
VMDUMP 0-END DCSS SYSTEM FORMAT M204/CMS
```

You can omit the DCSS operand if no saved segments are associated with the virtual machine. To enter a VMDUMP command from the service virtual machine console, prefix it with the Control Program command (#CP). Dump the associated spool files to magnetic tape using the Control Program SPTAPE command after the Model 204 dump files are created. Submit the tape and any supporting documentation, such as the Model 204 audit trail, to Technical Support for analysis.

Using unformatted system dumps (z/OS)

For large Onlines, in the event of an error that results in a snap, you may find that the amount of diagnostic information written to CCASNAP is excessively large and takes a long time to write. In such cases, z/OS users can use the system asynchronous dump option to take a high-speed, unformatted dump of the Model 204 address space. This is described in the *Model 204 Parameter and Command Reference* under the SNAPCTL parameter.

As an alternative, z/OS users can use an extended version of the SYSMDUMP facility to take up to ten unformatted dumps of the Model 204 address space. This is described in the following section. The asynchronous dump option is faster than the SYMDUMP option, which is faster than the CCASNAP option.

Allocating SYSMDUMP data sets

To allocate multiple SYSMDUMP data sets, insert one to ten DD statements with ddnames CCAMDMP0, CCAMDMP1, and so on. The following rules apply to the use of CCAMDMP data sets:

- Only empty CCAMDMP (0 – 9) data sets are selected for SYSMDUMP processing. At any time during the run, each data set contains at most one dump. This prevents SYSMDUMP processing from destroying or overwriting previous dump data sets.
- Use of tape data sets is not supported. The CCAMDMP data sets must be preallocated on disk devices.
- If the CCAMDMP allocations are DISP=SHR, then you can clear, copy, or analyze any of the data sets without bringing down the Model 204 Online.

- If an error occurs while allocating a CCAMDMP data set, allocation is not attempted again for the remainder of the run.
- If all CCAMDMP data sets are full, then SYSMDUMP processing terminates and either SYSUDUMP or SYSABEND is used for further ABDUMP processing, if either of these is specified in the JCL. Use of SYSUDUMP or SYSABEND statements in case of overflow is not recommended, however. These auxiliary data sets are more likely to fill up spool space than to provide important diagnostic information.
- CCAMDMP and SYSMDUMP DD statements are incompatible. If a SYSMDUMP statement is present, then the CCAMDMP statements are ignored, and multiple dump data sets cannot be created.
- Note that CCAMDMP data sets must be allocated on single extents.

Number of CCAMDMP data sets

The optimum number of CCAMDMP data sets is normally equal to the value of the SNAPLIM parameter, described in the *Model 204 Parameter and Command Reference*.

CCAMDMP data set DCB characteristics

DCB characteristics of CCAMDMP data sets must be the same as those recommended for the standard z/OS SYSMDUMP facility, shown in Table 14-2.

Table 14-2. CCAMDMP DCB characteristics

System	Organization	Record format	LRECL	BLKSIZE
z/OS	PS	FB or F	4160	4160

Space allocation

To estimate the space required for each CCAMDMP data set, add four megabytes to the size of the Model 204 address space. For example, a 20M address space needs 24M, or about 40 cylinders on a 3380 disk device.

Size of Model 204 address space

The address space actually used in a Model 204 run can depend on several factors in addition to the REGION parameter. For more information on this topic, refer to the IBM z/OS documentation.

Coordinating use of CCAMDMP and CCASNAP data sets

You can allocate both CCAMDMP and CCASNAP data sets in a Model 204 run. An advisable setting for CCASNAP is SNAPPDL. The CCASNAP data set is

then useful for quick diagnosis, while the CCAMDMP data sets provide more complete information.

Processing CCAMDMP data sets

You can process CCAMDMP data sets allocated in share mode using standard utilities such as IEBGENER. To archive and clear a dump data set, for example, copy it to tape, then copy a null file to the disk data set.

Using unformatted system dumps (z/OS)

Part IV

Recovery



This part describes the Model 204 system manager's role in tracking changes, system, and media recovery.

15

Checkpoints: Storing Before-Images of Changed Pages

In this chapter

- Overview of the Checkpoint facility
- Understanding the Checkpoint facility
- Overview of sub-transaction checkpoints
- Considerations for implementing sub-transaction checkpoints
- Implementing sub-transaction checkpoints in your job
- Before-image logging
- Obtaining checkpoint information (UTILC)

Overview of the Checkpoint facility

The Model 204 Checkpoint facility stores before-images of changed pages, or described another way, saves the image before applying the change to a page. Model 204 supports two types of checkpoints: transaction and sub-transaction.

- Transaction checkpoints are identical to what in pre-V6R1.0 Model 204 releases were simply called checkpoints. If a transaction checkpoint is not taken within a certain period because of too much update traffic, that is, times out, the Checkpoint facility tries again at the next interval.

- Sub-transaction checkpoints guarantee that a checkpoint is taken at the specified interval, which involves switching back and forth between transaction and sub-transaction checkpoints.

You can elect to take only transaction checkpoints by setting the CPTYPE parameter to 0, or you can take both transaction and sub-transactions by setting CPTYP to 1.

- Transaction checkpoints are saved to CHKPOINT, a sequential data set that contains copies of file pages before updates are applied (before-images, also called preimages) and marker records (checkpoints) that record the date and time when the system is quiescent (no updating activity).
- Sub-transaction checkpoints are saved to CHPNTS, the same type of sequential data set that saves the same information.

Model 204 would use either CHKPOINT and CHPNTS during recovery to restore (roll back) the original contents of updated files at a particular checkpoint, usually the most recent.

For more information

This chapter discusses the Checkpoint facility, preimage logging, and dumping the CHKPOINT file with the UTILC utility.

- HLI preimage logging is discussed in detail in the *Model 204 Host Language Interface Reference Manual*.
- “Recovery data sets and job control” on page 371 explains how to use CHKPOINT in the recovery process.
- Checkpoint data stream configurations are described in Chapter 15: “Checkpoints: Storing Before-Images of Changed Pages”.

Understanding the Checkpoint facility

Checkpoint markers and preimage copies of database pages are recorded on a sequential data set called CHKPOINT (and CHPNTS). Checkpoints taken during a Model 204 run mark times when no updates are in progress. Between checkpoints, before-images of Model 204 file pages are recorded in a checkpoint file as files are updated. After a system crash, you can roll back the database to its status at the time of a specific checkpoint. Changes made in the files between the time of that checkpoint and the time of the system crash are removed.

Checkpoints are taken on all files simultaneously to preserve logical file consistency. When file changes are rolled back, all files, regardless of their condition, are automatically restored to a checkpoint. Unless otherwise specified, files are rolled back to the most recent checkpoint.

By using the Checkpoint facility in conjunction with the Roll Back and Roll Forward facilities, you can recover a valid copy of the Model 204 database after a system failure.

Checkpoint parameters

You control the operation of the Checkpoint facility by specifying parameters on User 0's parameter line. Table 15-1 lists the checkpoint parameters.

Table 15-1. Checkpoint and recovery parameters

Parameter	Meaning
CPMAX	Maximum number of checkpoints saved during the run. The default is 32767.
CPTIME	Time in minutes between attempts to take automatic checkpoints. The default is 0.
CPTO	Amount of time in seconds allowed to quiesce updating User Language users, command users, and online IFAM jobs. The default is 0.
CPTQ	Amount of time allowed in seconds to quiesce updating batch Host Language Interface jobs. The default is 0.
CPTS	Time interval for in-flight transactions to complete before going into a sub-transaction wait.
CPTYPE	Choosing to use transaction-only checkpoints, or both transaction and sub-transaction checkpoints
CPSORT	Maximum number of times to retry an attempt to take a checkpoint initiated by the start or end of an IFAM2 job.
FRCVOPT	File recovery options. Settings indicate whether or not a file participates in checkpoint and/or roll forward logging.
NSUBTKS	Checkpoint facility uses three internal processes called pseudo subtasks (PSTs). The setting of this parameter must reserve at least three slots for checkpoint use, if checkpointing is active. Add another pseudo subtask for asynchronous checkpointing in an environment that supports 31-bit processing.
RCVOPT	Recovery options. Settings activate the Checkpoint facility and govern checkpoint and journal logging.

Taking a checkpoint

The process of taking a checkpoint is activated by one of the following events:

- Expiration of the specified time between checkpoints (CPTIME)
- CHECKPOINT command
- Host Language Interface IFCHKPT call
- Host Language Interface IFSTRT and IFFNSH calls

Note: Only the first IFSTRT call from an IFAM2 job activates the CHKPPST pseudo subtask. Subsequent IFSTRTs for the same IFAM2 job do not initiate checkpoint attempts.

Checkpoint algorithm

Once activated, the checkpoint algorithm proceeds as follows:

1. Prevents any new update units from starting until the checkpoint process is completed.
2. Waits a preset number of seconds for updating threads to quiesce.
The length of the wait depends on the type of threads that have active update units and the settings of the CPTQ and CPTO parameters.
 - If no update units are active, there is no wait.
 - If update units are active on batch Host Language Interface threads whose connection to the ONLINE region is initiated with an IFSTRT call, the task waits the number of seconds specified in the CPTQ parameter, or until all such threads have quiesced.
 - If update units are active on User Language threads or Host Language Interface threads whose connections to the ONLINE region are initiated with an IFDIAL call, the task waits the number of seconds specified in the CPTO parameter, or until all such threads have quiesced.
3. Times out the checkpoint if the specified number of seconds passes and update units are still active.
4. Writes a checkpoint record to the CHKPOINT file.
5. Allows new update units to begin after a checkpoint is taken or timed out.

If checkpoint has timed out, Model 204 attempts to take a checkpoint each time an update unit completes.

Taking asynchronous checkpoints (31-bit)

Taking asynchronous checkpoints is supported in operating environments with 31-bit processing. This reduces the amount of wait time for preimage writes to the checkpoint data set.

The pseudo subtask CHKPAWW performs the wait for preimage writes. If you are running in 31-bit mode, add one to the value of the NSUBTKS parameter. You might also need to increase the value of the LPDLST parameter.

Using the CHECKPOINT command

If a user issues the CHECKPOINT command to take a checkpoint, control returns immediately to the terminal of the user who issued the command. The terminal does not wait for the checkpoint to complete. When the checkpoint

process is complete, a message indicating the status of the checkpoint is issued to the terminal of the user who entered the command. If update units are active, control is returned to the user without taking a checkpoint.

Checkpoint processing with IFSTRT or IFFNSH calls

Whenever the CHKPPST pseudo subtask is activated by a Host Language Interface IFSTRT or IFFNSH call, a nonzero value specified on CPSORT causes the CHKPPST task to loop through the waiting process the number of times specified on CPSORT. New update units must wait for the entire length of the looping process.

For more information about the effect of CPSORT values on the CHKPPST task, refer to the *Model 204 Host Language Interface Reference Manual*.

Determining the status of a checkpoint

Any terminal user can issue the CHKMSG command to obtain a copy of the most recent checkpoint message. Host language users can issue an IFCHKPT call to determine the status of the most recent checkpoint. One of the following checkpoint status information messages is broadcast to the operator's console each time CHKPPST is activated (attempts to take a checkpoint):

```
*** M204.0843: CHECKPOINT COMPLETED ON date/time
```

or

```
*** M204.0843: CHECKPOINT TIMED OUT ON date/time DUE TO  
USER nn
```

Aborting a checkpoint

If a pending checkpoint causes too great a delay in new request initiation or host language jobs, a user with system manager privileges can issue the CHKABORT command to cause the pending checkpoint to time-out immediately. The CHKABORT command is issued without arguments.

M204CKPX checkpoint user exit

The checkpoint user exit, M204CKPX, can be invoked if linked in. This exit runs after the CHECKPOINT records are written, and just before the CHECKPOINT COMPLETED message is issued. All Model 204 databases are physically consistent on disk at this time, because all updated pages have been flushed to disk to prepare for checkpoint processing.

The exit runs *before* any update users are allowed to run in the Online, which allows users to write an exit that backs up all their database files, between the hours of x and y, all based on the user exit code.

When the exit abends or completes, the Online releases updating users to run and continues. Because this exit is invoked each time a checkpoint is taken, you must forego any extensive processing until the Online is in a low use period, then back up the files.

Example The following example is a M204CKPX ASSEMBLER exit. You can use the shell of the following program to write your own user exit. If your user exit abends, Model 204 tries to continue. All registers can be used. You can safely copy the files because the modified pages have all been flushed to disk and are physically consistent. The Online continues to service read-only users, and updating is suspended until this exit completes. You might want to include your own ESTAE exit macro to deal with abends.

A sample M204CKPX ASSEMBLER exit:

```

M204CKPX  CSECT
M204CKPX  AMODE 31
M204CKPX  TITLE 'TEST THE MODEL 204 CHECKPOINT USER EXIT'
X10      EQU    10
X11      EQU    11
X12      EQU    12
X13      EQU    13
X14      EQU    14
X15      EQU    15

        STM     X14,X12,12(X13)    SAVE CALLERS REGISTERS
        LR      X12,X15            ESTABLISH BASE REGISTER
        USING   M204CKPX,X12
        LA      X10,SAVEAREA       GET A(LOCAL REGISTER SAVEAREA)
        ST      X10,8(,X13)        CHAIN OUR SAVEAREA TO CALLERS
        ST      X13,SAVEAREA+4     CHAIN CALLERS SAVEAREA TO OURS
        LA      X13,SAVEAREA       SET A(OUR SAVEAREA)
        WTO     'M204CKPX CHECKPOINT EXIT INVOKED,
                UPDATERS SUSPENDED'

* ***** *
* * Add code here that calls routines to back up your * *
* * Model 204 databases. * *
* * * *
* DC      X'000000000000'          TEST TO SEE WHAT HAPPENS WHEN * *
*                                  THE USER EXIT ABENDS * *
* ***** *
        WTO     'M204CKPX EXIT ENDING, UPDATERS WILL BE RELEASED'
        L       X13,4(,X13)        RESTORE CALLERS SAVE AREA ADDRESS
        ST      X10,16(X13)        SET RETURN CODE (R15)
        LM      X14,X12,12(X13)    RESTORE CALLERS REGISTERS
        BR      X14                RETURN TO CALLER
        DS      0D
SAVE AREA DS 18F                    REGISTER SAVE AREA
        LTORG
        END

```

If Model 204 detects an abend in the M204CKPX exit, the following message is written to the JES log or is displayed on the z/VM console:

```

M204.CKPX: ABEND IN M204CKPX IGNORED, ATTEMPTING TO CON-
TINUE

```

Note: If M204CKPX abends, Model 204 attempts to continue processing, but there is no guarantee that it can do so. Further processing might be prevented for one of the following reasons.

- M204CKPX destroyed storage that Model 204 depends on.
- Left interrupts disabled, or did not restore the ESTAE or ESPIE macro routines.

Overview of sub-transaction checkpoints

A sub-transaction checkpoint is a checkpoint that can be taken while updating transactions are in progress and un-committed, eliminating checkpoint timeout situations. A sub-transaction checkpoint is a guaranteed checkpoint; it can be taken under almost all updating situations.

The exceptions are long-running file-update commands, which must run to completion before a transaction or a sub-transaction checkpoint can be taken. Examples of potentially long-running file-update commands, would be CREATE, INITIALIZE, and REDEFINE FIELD. If any of these commands require an extended period of time to complete, a sub-transaction checkpoint will be postponed. Generally, these commands are issued infrequently in production Onlines.

Guaranteed checkpoints improve recover ability, reduce recovery time, and further improve 24X7 availability.

Transaction checkpoints are identical to what in previous Model 204 releases were simply called checkpoints.

Reviewing transaction checkpoints

At the time a transaction checkpoint is taken, all updates have ended and been written to their respective M204 disk files. A transaction checkpoint is indicated by a date/time stamp record on the checkpoint, journal, and deferred update data sets, and represents a state to which a restart rollback can restore a set of Model 204 files, in preparation for using roll forward to reapply later updates.

For a transaction checkpoint to be taken, user threads must end their update units. This can be done with some form of COMMIT or IFCHKPT.

Introducing sub-transaction checkpoints

At the time a sub-transaction checkpoint is taken, all updates will not have ended (although a non-zero value of CPTS will give them time to do so and therefore change a sub-transaction checkpoint attempt into a transaction checkpoint).

A typical update unit is made up of a sequence of sub-transactions. A typical sub-transaction consists of reading a Model 204 page to be updated, writing the image of the page to the checkpoint stream, making an update to the page,

creating a back out and constraint entries in CCATEMP (in case the update is backed out) and writing the update to the journal stream.

At the time a sub-transaction checkpoint is taken, all sub-transactions will have ended and been written to their respective Model 204 disk files. However, there will still be active update units with associated non-zero back out and constraint entries. These are written to the checkpoint stream. So, a sub-transaction checkpoint is indicated by a date/time stamp record on the checkpoint stream, along with all back out and constraint entries for active updates. This is followed by a sub-transaction checkpoint completion record. The checkpoint stream therefore contains a snapshot of active update units at the time of checkpoint.

For sub-transaction checkpoints only the initial date/time stamp record is written to the journal stream and deferred update data sets.

Using sub-transaction checkpoints in recovery

At restart recovery, the back out and constraint logs from the input restart streams are used to recreate the update environment needed to recover transactions interrupted by the original sub-transaction checkpoint.

Sub-transaction enabled Onlines use two checkpoint streams: CHKPOINT and CHPNTS. At any given time during the Online run, one stream or the other is the current checkpoint stream—the one to which preimages are written.

A sub-transaction checkpoint causes the current stream to be switched and checkpoint information written to the beginning of the new stream. If the Online comes down before the sub-transaction checkpoint completes then it is unusable and the previous good checkpoint must be used for restart recovery.

CCATEMP, NDIR, and NFILES must be the same size or greater than the settings used in the run being recovered.

Using transaction or sub-transaction checkpoints

If a transaction checkpoint is taken on an sub-transaction enabled Online then the current checkpoint stream is rewound instead of switched.

Note: The initialization and termination checkpoints are always transaction checkpoints. The command CHECKPOINT E Q forces the next checkpoint to be a transaction checkpoint. The checkpoint at the end of a roll forward is always a transaction checkpoint.

IFAM SIGNON/SIGNOFF and IFCHKPT checkpoints are always transaction checkpoints. And, there are certain run configurations and checkpoint stream/and file type configurations, which require all checkpoints to be transaction checkpoints. This means that sub-transaction checkpoints cannot be enabled for those configurations. An example of this would be all NUSER=1 runs.

Considerations for implementing sub-transaction checkpoints

Sites that find they regularly have checkpoints timeout, with no intervening successful checkpoint over a significant duration of time, might consider this option.

The best way to rectify this situation is to attempt to modify the applications that cause checkpoints to timeout by improving the use of the COMMIT statement. However, this may not be a reasonable approach given time and resource constraints or the nature of the application.

An Online with sub-transaction checkpoints enabled uses multiple KOMMs. KOMMOPT is no longer an MP-only parameter.

Implementing sub-transaction checkpoints in your job

The CPTYPE parameter designates the type of checkpoints to use.

- CPTYPE=0, the default, indicates that transaction checkpoints are activated.
- CPTYPE=1 indicates that both transaction and sub-transaction checkpoints are activated.

Note: CPTYPE is resettable by the system manager.

Requirements for CPTYPE=1

To enable sub-transaction checkpoints, you must make the following adjustments to your data sets and Model 204 parameters. Otherwise the run terminates with one of the M204.2684 messages, for example:

M204.2684 CHECKPOINT CONFIGURATION CONFLICT

- The data set CHPNTS must be defined to the job, in addition to CHKPOINT. Rocket Software recommends that you allocate the CHPNTS data set with the same space parameters as the CHKPOINT data set.

If the CHPNTS data set is not defined or cannot be opened, an error message is issued during initialization and the job is terminated.

- The NUSERS parameter must be greater than one. If NUSERS equals one, then any stream definition for CHPNTS is ignored and the run is not a sub-transaction enabled run.
- The KOMMOPT parameter must be equal to one.
- If you set CPMAX to greater than one, the following message is issued, and Model 204 resets CPMAX to one:

M204:2685 CHPNTS IS OPEN SO CPMAX SET TO 1

- The DKUPDTWT parameter must be set to zero, which is the default.
- You can set the CPTS parameter to establish the number of seconds to delay the start of new updating transactions. A non-zero CPTS time interval gives in-flight transactions a chance to end before they are forced into a sub-transaction wait.

If all transactions end during the CPTS interval, a transaction checkpoint is taken instead of a sub-transaction checkpoint.

- The CPTIME parameter is still required to establish the time interval, in minutes, between automatic checkpoints. It must be greater than zero.
- Set RCVOPT to nine to enable checkpoints and journals

CPTO and CPTQ can be left in your CCAIN stream even if you set CPTYPE=1. However, they will be ignored unless CPTYPE is reset to 0.

As long as all data sets and parameters are appropriately defined and requirements met within your job, you can reset back and forth between CPTYPE=0 and CPTYPE=1.

Checkpoint definition restrictions for sub-transaction checkpoints

Both CHKPOINT and CHPNTS must be present. Each may be defined as a stream but may *not* contain:

- A ring stream
- More than 16 levels of recursive stream definitions
- A CMS formatted file

Before-image logging

Every updated database page is copied (logged) into the CHKPOINT data set before any change is performed. Whenever a request to update a page is received, Model 204 first checks to see if the page was logged since the last checkpoint. If the page has not been logged, the page is logged and the update allowed to proceed. If the page has already been logged, the update proceeds immediately.

You can disable logging of before-images on a file-by-file basis by using the file parameter FRCVOPT. If logging is disabled, the file can be recovered only by using the media recovery procedures. (See “Media recovery NonStop/204” on page 388 for more information on media recovery.)

Creating the CHKPOINT/CHKPNT (and CHPNTS) data set

The checkpoint data set is named CHKPOINT in z/OS and z/VM; in z/VSE, use the name CHPNT. If your site uses the sub-transaction capability as well, that checkpoint data set is named CHPNTS for all operating systems.

Notes:

Due to changes in Journal record layouts, CCAJRNL and CHKPOINT/CHKPNT data sets are not compatible with previous releases of Model 204. For details, see “Journal block header information for SWITCH STREAM” on page 289.

If your site has sub-transaction checkpoints enabled, you must create a CHPKNTS data set in addition to the original checkpoint data set. The following discussion applies equally to creating a CHPKNTS data set.

To create a checkpoint data set:

1. Include a CHKPOINT (or CHPKNT, and as needed, CHPKNTS) DD statement in the JCL.
2. Set RCVOPT to include the X'01' bit on User 0's parameter line.

A checkpoint data set is a sequential, unblocked data set with a record length equal to the Model 204 page size (6184 bytes). You can store it only on disk.

You may define the CHKPOINT and CHPKNTS data sets using parallel streams. The only restriction on CHKPOINT and CHPKNTS stream definitions is that they may not contain CMS data sets. There are no restrictions on the type of stream you can use to define the CCAJRNL and CCAJLOG journals.

Calculating disk space required

You can determine amount of disk space required by the CHKPOINT data set in any given run by using the high water mark of records (page preimages) written to the CHKPOINT data set. You can obtain the high water mark via a User Language request. For example, you may want to include the following procedure as part of your CCAIN input stream just prior to EOJ:

```
BEGIN
PRINT 'HIGH WATER MARK FOR CHECKPOINT RECORDS' AND -
      $CHKPINF(8)
END
```

Calculate the maximum amount of disk space in use in this run in the CHKPOINT data set by dividing the high water mark of records (page preimages) written to the CHKPOINT data set by the number of pages per track for your device type. To determine pages per track, see the *Rocket Model 204 File Manager's Guide*.

You can also monitor the size of the CHKPOINT data set using any existing utility appropriate for your operating system.

Generous sizing of the CHKPOINT data set is important, because Model 204 terminates whenever the CHKPOINT data set becomes full. For this reason, make the primary extent at least 50% greater than that calculated above. Further, define the data set with a substantial secondary extent, say 25% of the primary extent, and 15 extents of this size should be guaranteed available. For

information about configuring the CHKPOINT data set, see Chapter 17, “Configuring Checkpoint and Journal Data Streams”.

Maintaining single volume CHKPOINT data sets

You cannot process multivolume CHKPOINT data sets during recovery because of limitations of the BSAM access method used by RESTART recovery.

If a CHKPOINT data set is written to multiple volumes, because of secondary allocation under z/OS, then it is necessary to copy this file onto a single volume before running RESTART recovery.

Avoid multivolume CHKPOINT data sets by making sure that primary disk allocation is adequate, or by setting the CPMAX parameter (explained as follows, “Limiting the size of CHKPOINT”) to a sufficiently low value.

Limiting the size of CHKPOINT

Model 204 writes to the CHKPOINT data set until the specified maximum number of checkpoints saved (CPMAX) are taken and the next checkpoint is about to be taken. At that point, Model 204 rewinds the CHKPOINT data set and takes the next checkpoint. The data previously written to CHKPOINT is overwritten.

If you set CPMAX to one and CPTIME to less than 15, then the CHKPOINT data set will generally be small enough to fit on a single volume and, usually, occupy no more than about 20 cylinders. CPMAX set to one ensures that you keep only the most recent checkpoint and the preimages logged since that checkpoint. When the CPMAX-plus-one checkpoint is about to be taken, the CHKPOINT data set rewinds and the new checkpoint is taken again at the beginning of the data set. This keeps the data set small and ensures faster recovery.

To preserve all checkpoints and all preimages in the CHKPOINT data set, omit CPMAX on User 0's parameter line.

Obtaining checkpoint information (UTILC)

The UTILC utility is a standalone batch utility that provides information about the Model 204 checkpoint process by interpreting a checkpoint file and printing out checkpoint records in dump format. UTILC is a debugging aid for Model 204 users working with Technical Support staff on a specific problem. However, the information provided by UTILC can also be useful for performance analysis and capacity planning.

Data in a checkpoint data set consists of the record types shown in Table 15-2.

Table 15-2. Checkpoint record types

Type code	Contents
1 (01)	File page preimage
2 (02)	Checkpoint
3 (03)	Copy of the roll forward file directory
4 (04)	Unused
5 (05)	File Parameter List (FPL) page preimage
6 (06)	Unused
7 (07)	Batch (IFAM2) job sign on
8 (08)	Batch (IFAM2) job sign off
9 (09)	Deferred update open
10 (0A)	Discontinuity
11 (0B)	RESET FISTAT (turn off physically broken bit)
12 (0C)	Dynamic allocation

Checkpoint record formats

In checkpoint records (type 2) the date and time of the checkpoint is stored in the first 8 bytes of the record, offset 0(0), in the format:

00yydddFhhmmssth

Where

- *yyddd* is the Julian date
- *hhmmssth* is the time

For example, *88.335 15:04:27.88* is represented as *0088335F15042788*.

Each record ends with a 40-byte trailer that includes the 1-byte numeric type code at offset 6172 (181C) followed by the date and time of the posted checkpoint in the format:

yydddFxxxxxxxx

Where

- *yyddd* is the Julian date
- *xxxxxxxx* is the millisecond count of the day

Trailer information identifies preimages, record types 1 and 5:

- First 8 bytes of the trailer, offset 6144(1800) from the start of the record, contains the name of the file in EBCDIC.

- Last 4 bytes of each preimage record, offset 6180(1824) from the start of the record, is a page identifier in the format:

TTPPPPPP

Where

- *TT* is a 1-byte table identifier:

Identifier code	Table
00	File Control Table (FCT)
01	Table A
02	Table B
03	Table C
04	Table D

- *PPPPPP* is a 3-byte hexadecimal table-relative page number.

UTILC input data sets

You can use only single data sets as input to the UTILC utility. You must process members of concatenated, parallel, or ring streams one at a time.

UTILC options

The following options control the execution and output of UTILC:

- FROMDATE and TODATE specify the range of dates for which checkpoint records are printed. FROMDATE and TODATE must be 5-digit Julian dates (for example, 88062).
- FROMTIME and TOTIME specify the time range for which checkpoint records are printed. FROMTIME and TOTIME must be in 24-hour clock (hhmmss) format (for example, 151003 = 3:10:03 PM).
- RECTYPE specifies the type of records to be printed. For a list of record types, see Table 15-2 on page 343. The default is all record types.

Repeat RECTYPE as necessary. The following example prints all preimages and all File Parameter List page preimages.

RECTYPE=1 , RECTYPE=5

- FILENAME specifies the ddname of the file, the name specified in the CREATE command, to be printed. Pages without file names are also printed unless excluded by record type; for example, CHECKPOINT records.

- FROMPAGE and TOPAGE specify the range of Model 204 file pages printed. Page numbers must be hexadecimal. The following example prints all Table B pages for all files:

```
FROMPAGE=02000000, TOPAGE=02FFFFFF
```

- Use the FILENAME option to further restrict printing to selected pages from specific files.
- TRAILERS=ONLY specifies printing only page trailers.

If you do not specify the TRAILERS=ONLY parameter, entire checkpoint records are printed.

UTILC examples

The following examples provide the job control statements to run UTILC in z/OS, z/VSE, and z/VM environments. Each example shows how to request printing of checkpoint record trailers for the hour between 12:05:13 and 13:05:13 in different operating systems.

z/OS

```
//UTILC EXEC PGM=UTILC, PARM=' FROMTIME=120513, TOTIME=130513,
//          TRAILERS=ONLY'
//STEPLIB DD DSN=M204.LOADLIB, DISP=SHR
//CCAPRINT DD SYSOUT=A
//SYSU DD SYSOUT=A
//CHKPOINT DD DSN=M204.CHKPOINT, DISP=OLD
```

z/VSE

The input stream, CHPNT, for running UTILC is defined in the JCL by DLBL and EXTENT statements or a TLBL statement:

```
// JOB UTILC PRINT DISK CHECKPOINT FILE
// DLBL CHPNT, 'M204.CHKPOINT.FILE'
// EXTENT SYS001, SYSWK1, , 1390, 1000
// EXEC UTILC, SIZE=AUTO, FROMTIME=120513, TOTIME=130513,
//          TRAILERS=ONLY
//&
```

If the input file is on tape, assign the symbolic unit SYS004 to the tape drive on which the tape is mounted. The output is printed on the symbolic unit SYSLST:

```
// JOB UTILC PRINT TAPE CHECKPOINT FILE
// TLBL CHPNT, 'M204.CHKPOINT'
// ASSGN SYS004, X'300'
// EXEC UTILC, SIZE=AUTO, FROMTIME=120513, TOTIME=130513,
//          TRAILERS=ONLY
//&
```

z/VM

The following UTILC EXEC procedure runs UTILC. The command format is:

```
UTILC dsn mode (FROMTIME hhmmss TOTIME hhmmss  
TRAILERS=ONLY
```

Where

- *dsn* is the data set name.
For z/OS format data sets, the DSN is specified with spaces instead of periods between the qualifiers.
For CMS data sets, the DSN is specified as *filename filetype*.
- *mode* specifies the access mode of the disk containing the file to be processed.
- For tape files, the DSN and MODE parameters are replaced by the keyword TAPE.

Tape files also require issuing a FILEDEF, and a LABELDEF, if necessary, for the CHKPOINT data set before issuing the UTILC command. For example:

```
FILEDEF CHKPOINT TAP1 SL VOLID 12345 (RECFM FB LRECL  
6184 BLKSIZE 6184  
LABELDEF CHKPOINT standard labeldef parameters  
UTILC D TAPE  
(FROMTIME 120513 TOTIME 130513 TRAILERS=ONLY
```

Note: The length of optional parameters you can specify is limited by the z/VM console line length of 130 characters.

16

System and Media Recovery

In this chapter

- Overview
- Recovery features
- Understanding transaction back out
- ROLL BACK facility
- ROLL FORWARD facility
- RESTART recovery considerations for sub-transaction checkpoints
- Restarting after a system failure
- Reporting recovery status
- Recovery data sets and job control
- Handling recovery failures
- Automated secondary recovery: reuse JCL
- Recovering deferred update mode files
- Recovering dynamically allocated data sets
- Media recovery
- Media recovery NonStop/204
- Using the UTILJ utility

- Using UTILJ to analyze problems
- Using the MERGEJ utility

Overview

Hardware, software, or media failures compromise database integrity. The recovery facilities of Model 204 detect and correct integrity problems that can affect some or all Model 204 files in Online or Batch updating jobs.

This chapter discusses individual transaction and database recovery facilities provided by Model 204 in the event of application, system, or media failure. However, the importance of a reliable, well-maintained backup plan cannot be overemphasized; see “Preparing for media recovery” on page 349

This chapter describes the User Language statements, Model 204 commands and parameters that are coordinated and used by the recovery facilities. The recovery facilities include:

For system recovery:

- Transaction Back Out
- RESTART recovery, which utilizes
 - ROLL BACK processing
 - ROLL FORWARD processing

For media recovery

- NonStop/204 for file backups
- REGENERATE and REGENERATE ONEPASS commands processing for applying updates which have occurred since the file's last backup
 - ROLL FORWARD processing
 - UTILJ utility
 - MERGEJ utility

Other topics related to Model 204 recovery are discussed in the following chapters:

Topic	Discussed in...
Keeping an audit trail by recording and printing system activity	Chapter 13
Recovering application subsystem definitions (CCASYS)	Chapter 8
Storing preimages of changed pages and checkpoints	Chapter 18
Handling CCATEMP, the system scratch file, after a system crash	Chapter 5

Topic	Discussed in...
Keeping an audit trail by recording and printing system activity	Chapter 13
Using the system statistics	Appendix A

Consulting other Model 204 manuals

You can also consult:

- *Model 204 Parameter and Command Reference* for:
 - The CHECKPOINT command can request either a transaction checkpoint or a sub-transaction checkpoint. The options are TRAN, the default, for a transaction checkpoint and SUBTRAN for a sub-transaction checkpoint.
 - The MONITOR command has a SUBTRAN option to display the status of sub-transaction checkpoints.
- *Model 204 File Manager's Guide* for:
 - Error diagnosis
 - Methods for maintaining transaction efficiency
 - Additional information about roll forward file types
 - Discussion of update units
- *Model 204 Host Language Interface Programming Guide* and *Model 204 Host Language Interface Reference Guide* for:
 - Details regarding Host Language Interface calls, IFAM1, IFAM2, and IFAM4 applications
 - Transaction behavior
- *Model 204 User Language Manual* for:
 - Maintaining transaction efficiency

Preparing for media recovery

To use media recovery effectively, an installation must establish procedures for backing up files on a regular basis and for archiving and maintaining CCAJRNL files that contain a complete history of file activity.

Media recovery requires the following preparation:

- Install the MERGEJ utility.
Installation and link-editing of MERGEJ is discussed in the installation guide for your operating system.
- Dump Model 204 files on a regular basis.

Use the Model 204 DUMP command or any other DASD management utility.

Note: If your Model 204 backups are not done using a Model 204 DUMP command, you must make certain that no one is updating the Model 204 file at the same time the backup is being performed. If you use a backup from a Model 204 dump file, updates made to the file since the dump are reapplied by the ROLL FORWARD facility of the REGENERATE command using single, concatenated, or merged journals as input.

- Produce and archive journals. You must maintain a complete record of update activity in the form of archived CCAJRNLS, in all types of Model 204 runs (Online, Batch, and RESTART recovery).

Note: In a z/VSE environment, the use of concatenated journals for regeneration is not supported.

- Run the UTILJ utility against any journal that does not have an end-of-file marker before using the MERGEJ utility. See “Using the UTILJ utility” on page 393.
- Merge any overlapping journals needed for a media recovery run by running the MERGEJ utility.

MERGEJ is discussed in “Using the MERGEJ utility” on page 401.

- If you are running media recovery, you must ensure that all files that were in deferred update mode during part or all of the time period you specify in the REGENERATE command are still in deferred update mode during the media recovery run. This might require larger deferred update data sets than those in the original run.

Recovery features

Model 204 protects the physical and logical consistency of files at all stages of processing using the following features.

Handling error diagnosis

Error diagnostic features, described in the *Model 204 File Manager's Guide*, ensure data integrity by command syntax checking, compiler diagnostics, error recovery routines for data transmission from user terminals, and maintaining and checking trailers on each table page.

Handling Transaction Back Out

Logical consistency of individual updates to a Model 204 file is protected by the Transaction Back Out facility, which is activated by the FRCVOPT and FOPT parameters. Transaction back out locks records until an update is completed.

If the update is not completed, the effect of the incomplete transaction on file data is automatically reversed.

Handling a system recovery

Model 204 system recovery procedures involve the following functions:

Facility	Description
Checkpoint facility	Logs <i>preimages</i> of pages, which are copies of pages before changes are applied, to the CHKPOINT data set.
Journal facility	Contains incremental changes that reflect updates to database files; see Chapter 13: “Tracking System Activity (CCAJRNL, CCAAUDIT, CCAJLOG)”
RESTART recovery	In the event of a system failure, the restart facility is invoked by the RESTART command. This invokes the ROLL BACK facility and, optionally, the ROLL FORWARD facility.
ROLL BACK facility	Rolls back all Model 204 files to a particular checkpoint—that is, undoes all of the changes applied to the files since the last checkpoint.
ROLL FORWARD facility	Rolls each file forward as far as possible—that is, reapplies all complete updates and committed transactions that were undone by the ROLL BACK facility.

System recovery is initiated by the RESTART command, which invokes either the ROLL BACK facility, the ROLL FORWARD facility, or both.

In the event of a system failure, you can restore all files. Any Model 204 file that was being updated at the time of a system failure is flagged. Subsequently, whenever the file is opened, the following message is issued:

```
M204.1221: filename IS PHYSICALLY INCONSISTENT
```

Once recovery procedures are executed, any discrepancies in the files are corrected and the message is no longer issued.

Handling a media recovery

In the event of media or operational failure, you can restore individual files from previously backed-up versions. Restoration of the files and reapplication of the updates are invoked by the REGENERATE and REGENERATE ONEPASS commands.

When recovery procedures are completed, status reports are produced for each file participating in the recovery.

Understanding transaction back out

Transaction back out, also called TBO, protects the logical and physical consistency of individual transactions. Individual updates performed by User Language statements or corresponding Host Language calls can be reversed automatically or explicitly, when:

- A request is canceled
- A file problem or runtime error is detected
- A user is restarted
- Requested by a User Language or Host Language Interface program via the BACKOUT statement or IFBOUT call respectively

Transaction back out is enabled on a file-by-file basis by specifications made on the FRCVOPT and FOPT file parameters.

Making updates permanent

A **commit** is the process of ending a transaction and making the transaction updates permanent. A commit is performed by issuing:

- User Language COMMIT statement
- User Language COMMIT RELEASE statement
- Host Language Interface IFCMMT call

Any event that ends a transaction such as end-of-request processing is an implicit commit.

Transaction boundaries

For User Language updates, a transaction begins with the first evaluation of an updating statement in a given request. One request can execute multiple transactions.

User Language transactions are ended by any one of the following events:

- A BACKOUT statement, the transaction is ended after the back out is complete
- A COMMIT statement
- A COMMIT RELEASE statement
- End of a subsystem procedure, unless overridden in the subsystem definition
- A CLOSE FILE command or any file updating command, such as RESET or REDEFINE, following an END MORE command

- Return to terminal command level using the END command or, for nested procedures, the end of the outermost procedure

In an application subsystem environment, also called Subsystem Management facility, you can make transactions span request boundaries by using the AUTO COMMIT=N option in the subsystem definition.

Once a transaction ends, another transaction begins with the next evaluation of an updating statement.

Updates that can be backed out

For files that have transaction back out enabled, you can back out all User Language or Host Language Interface updates that are not yet committed. You can initiate back out only on active or incomplete transactions.

Updates that cannot be backed out

The updates that you cannot back out include:

- The following commands:

ASSIGN
BROADCAST FILE
CREATE
DEASSIGN
DEFINE
DELETE
DESECURE
INITIALIZE
REDEFINE
RENAME
RESET
SECURE

- The following Host Language Interface calls:

IFDFLD
IFDELF
IFINIT
IFNFLD
IFRFLD
IFRPRM
IFSPRM

- Procedure definitions (PROCEDURE command)
- Procedure edits
- Any file update that has been committed
- Any update to a file that does not have transaction back out enabled

See the *Rocket Model 204 Host Language Interface Reference Manual* for a discussion of Host Language Interface transaction behavior.

Back out mechanism

Transaction back out logically reverses transactions by applying compensating updates to files participating in the transaction. For example, the logical reversal or back out of a STORE RECORD statement is equivalent to processing a DELETE RECORD statement.

Compensating updates for incomplete transactions are stored temporarily in a back out log in CCATEMP and then deleted when the transaction is committed or backed out.

Lock-pending-updates locking mechanism

Transaction back out utilizes an additional locking mechanism, called lock-pending-updates, to assure data integrity and transaction independence. A record in the process of being updated in one transaction is inaccessible by other transactions until the updating transaction commits or is backed out.

The lock-pending-updates (LPU) lock is separate from other record locks and can be enabled without transaction back out. The LPU lock is:

- Always exclusive
- Placed on each record as soon as it is updated by a transaction
- Held until the transaction commits
- Released by:
 - User Language BACKOUT, COMMIT, and COMMIT RELEASE statements
 - Host Language Interface IFBOUT, IFCMMT, and IFCMTR calls
 - All implied commits or automatic back outs of the transaction

Note: The User Language RELEASE RECORDS statement does not release the LPU lock.

Back out and constraint logs

A back out log of compensating updates and a constraints log, which assures availability of resources freed by active transactions, are built in CCATEMP and freed when the update ends or is backed out. Sufficient storage space must be allocated for CCATEMP to accommodate the back out and constraints logs. Inadequate space can terminate the run.

Understanding the back out log

All pages not used by any other operation in CCATEMP can be used for back out logging. A minimum of one page is required.

The number of pages needed in CCATEMP depends on the number of concurrent transactions and the size of each transaction. More information about calculating CCATEMP space is given in “Sizing CCATEMP” on page 146.

Rocket Software recommends that you keep transactions short with frequent COMMIT statements. Short transactions minimize the CCATEMP space used, allow more frequent checkpoints, reduce record contention from lock-pending-updates locking, and reduce overhead for all updating users by keeping the back out and constraints logs small.

Understanding the constraints log

The constraints log is a dynamic hashed database that resides in a few virtual storage pages with any extra space needed residing in CCATEMP. In this database there are primary pages and overflow pages, also called extension records. When a primary page fills up, additional records go on to overflow pages that are chained from the primary page. When a primary page become too full, a process is evoked which splits the page into two pages. This process continues to the maximum-allowed number of primary pages at which point the splitting process stops and the overflow page chain is allowed to grow as large as required.

As the constraints database contracts, this process is reversed. Those pages with only a few records are merged together or compacted. This process continues until only one page remains.

The resources freed by User Language statements, such as Table B space, record numbers, and Table C hash cells, cannot be reused by other active transactions until the transaction that freed the space ends. Before constrained resources can be used, any new transactions needing those resources must search the constraints log.

The constraints log uses specially formatted pages with approximately 300 records per page. CCATEMP space considerations for constraint logging are the same as for logged back out images, which is described in “Understanding the back out log”.

All CCATEMP pages not used by any other operation are available for constraints logging. A minimum of one page is required.

Optimizing the constraints log performance

The parameters CDMINP2X and CDMAXP2X optimize the constraints log performance. The constraints log is used in transaction back out. The

constraints log uses several pages that are permanently allocated in storage; when those pages become full, additional CCATEMP pages are allocated.

The number of permanently allocated pages is controlled by the CDMINP2X parameter. Allocating more pages in storage saves time, because disk read-writes are not necessary and CPU time spent accessing pages is lower.

Note: However, the permanent allocation of too many pages (thousands) may waste the virtual storage, because unused pages may not be reused while unused CCATEMP pages may be reused for other purposes.

Model 204 uses the constraints log efficiently by merging or splitting pages to keep amount of data per page within certain limits. This saves space used by the constraints log, but requires additional CPU time. The CDMAXP2X parameter specifies the maximum number of pages to keep in compacted form.

Note: When the number of pages used in the constraints log exceeds CDMAXP2X setting, additional pages are not compacted. Compaction saves space and in some cases CPU time for finding records in the constraints log, but requires additional CPU time for the compaction itself. Using too many compacted pages (thousands) may negatively affect performance.

Disabling transaction back out

By default, transaction back out is enabled for all Model 204 database files.

You can disable transaction back out and lock-pending-updates using the following file parameter settings.

To disable...	Set...
Back out mechanism	FRCVOPT X'08' bit on
Lock-pending-updates	FOPT X'02' bit on

You can specify these settings when the file is created or with the RESET command.

Because lock-pending-updates are required for transaction back out, setting the FOPT X'02' bit on automatically turns on the FRCVOPT X'08' bit.

Disadvantages to disabling transaction back out

The following considerations apply to transaction back out:

- If transaction back out is disabled, you cannot back out updates.
- Transaction back out provides a mechanism for recovery from file-full conditions.
- You cannot define fields with the UNIQUE or AT MOST ONE attribute in files that do not have transaction back out enabled.

The UNIQUE attribute is described in the *Model 204 Parameter and Command Reference* and the *Model 204 File Manager's Guide*.

ROLL BACK facility

The ROLL BACK facility is activated by the ROLL BACK argument of the RESTART command. ROLL BACK processing removes changes applied to files by restoring copies of file pages as they existed before updates, called preimages, from the CHKPOINT data stream. All preimages logged after the last or specified checkpoint are restored.

The ROLL BACK facility, in conjunction with the ROLL FORWARD facility, described in “ROLL FORWARD facility” on page 359, is used for recovery of the entire database when:

- Abnormal termination occurs in Online, Batch 204, IFAM1 or IFAM4

Note: For IFAM1 only the ROLL BACK facility is available; the ROLL FORWARD facility is not supported for IFAM1

- Operating system or partition abends

ROLL BACK processing

The CHKPOINT stream created in the failed run is processed as input and called the RESTART stream during ROLL BACK processing, which consists of two passes through the RESTART stream.

For maximum efficiency, CCATEMP should have at least 33 pages.

ROLL BACK processing, Pass 1

In Pass 1, ROLL BACK processing scans the RESTART stream from the beginning, looking for the last checkpoint taken in the failed Online or for the checkpoint, if specified, in the RESTART command.

For each file in deferred update mode, the deferred update stream is opened and repositioned at the appropriate checkpoint record.

1. The system builds a directory of files opened in deferred update mode or changed after the specified checkpoint. IDs of gaps in recovery information of a particular file (discontinuities) that occurred after the last checkpoint are also noted in the file directory.

A **discontinuity** is an event that overrides all other file update events and cannot be rolled across. For example, if you add two new records to a file, close the file, and then issue a CREATE command on that file, it does not matter that new records were added. They were destroyed when the file was created. If you run recovery across this discontinuity, recovery notices the discontinuity and does not apply the two new records to the file. Recovery ignores them as useless.

The following processes cause discontinuities: CREATE FILE, CREATEG, INITIALIZE, a file updated by a second job, RESET FISTAT, RESTORE, RESTOREG, RESET FRCVOPT, REGENERATE, ROLLBACK; ROLL FORWARD, and system initialization.

2. When the end of the checkpoint stream is reached, the files named in the file directory are opened. The NFILES and NDIR parameter settings must be large enough to accommodate these opens.
3. After the files and deferred update streams are opened, a list of files that you cannot recover and a list of files that you can roll back are displayed on the operator's console.
4. Files that cannot be recovered will be listed following the message:

M204.0145: THE FOLLOWING FILES CANNOT BE RECOVERED

This table lists the possible reasons:.

Reason for failure	Meaning
DEFERRED UPDATE CHKP MISSING	The scan of the deferred update data set did not locate the checkpoint used for ROLL BACK processing.
DEFERRED UPDATE READ ERROR	The scan of the deferred update data set cannot be completed because of a read error.
DEFERRED UPDATES MISSING	Deferred update data set cannot be opened.
MISSING	File cannot be opened.
ROLL BACK INFORMATION IS OBSOLETE	File was modified by a Model 204 job after the last update in the run being recovered.
IS ON A READ-ONLY DEVICE	(CMS environment only) File is on a device for which Model 204 has read-only access.

5. If a list of files that cannot be recovered is displayed, the ERROR clause of the ROLL BACK portion of the RESTART command is examined.

ERROR clause	Means
ERROR CONTINUE	The listed files are ignored and recovery proceeds for other files.
ERROR STOP	The recovery run terminates. Errors can be corrected and recovery run again.

ERROR clause	Means
ERROR OPERATOR or no ERROR clause	<p>The message <code>SHOULD RESTART CONTINUE?</code> is displayed on the operator console.</p> <ul style="list-style-type: none"> YES (Y) response has the same effect as an ERROR CONTINUE clause. NO (N) response has the same effect as an ERROR STOP clause.

- After the list of files that you can roll back is displayed, you can open any unlisted files for retrieval.

ROLL BACK processing, Pass 2

In Pass 2, ROLL BACK processing reapplies page images to the files by reading the CHKPOINT stream backwards.

ROLL BACK processing stops at the specified checkpoint and notifies the operator. Files with discontinuities are not rolled back beyond the discontinuity. If ROLL FORWARD processing is not used, the files are closed and made available for general use.

ROLL FORWARD facility

If ROLL BACK processing was successful, the ROLL FORWARD facility is activated by the ROLL FORWARD argument of the RESTART command. ROLL FORWARD processing reapplies the file updates from the recovery data set CCARF to files rolled back by ROLL BACK processing. CCARF is the journal file, CCAJRNL, from the run being recovered.

If ROLL BACK processing is successful, the CCARF journal stream (CCAJRNL from the run being recovered) is opened.

The ROLL FORWARD facility is available for Online, Batch 204, and IFAM4 jobs.

RESTART ROLL FORWARD recovery is version specific

ROLL FORWARD recovery is *not* compatible with Model 204 journals from previous releases. If you attempt ROLL FORWARD processing using a journal created under a previous release, processing is terminated with the following error:

```
M204.2501: RELEASE INCOMPATABILITY
```

ROLL BACK processing remains compatible with previous releases. Rocket Software recommends that you backup all files prior to installing the current release. Also, since the format of all update journal records is now eight bytes larger than in previous releases, the journal may need additional space.

The amount of data lost due to a system crash is minimized by the journal buffer being forced out upon the completion of every update unit (COMMIT). An update unit is a sequence of operations allowed to update the database.

After the operator is notified that ROLL FORWARD processing is complete, terminal users can open the recovered files for updating.

Logging ROLL FORWARD processing

To enable ROLL FORWARD logging in the Online, the following requirements must be met:

- CCAJRNL output stream must be present
- SYSOPT setting must include the 128 option
- ROLL FORWARD logging option must be set on the RCVOPT parameter (X'08')
- CHKP and JRIO object modules must be present in the link-edit of the Model 204 program to be run
- To control ROLL FORWARD logging on a file basis, use the FRCVOPT file parameter, described in the *Model 204 File Manager's Guide*.

Message for the Operator

While ROLL FORWARD is processing, the following message is issued:

```
M204.1992: RECOVERY: PROCESSING ROLL FORWARD BLOCK#  
blocknumber date-timestamp
```

The message gives the processing sequence number and date-time stamp. It is issued at the Operator's console for each occurrence of an hour change in the date-time stamp of the CCARF update records.

You could receive a maximum of 24 such messages in a 24-hour period.

ROLL FORWARD processing

ROLL FORWARD processing consists of one pass of CCARF and performs the following steps:

1. Reads the journal stream, CCARF, starting from the ROLL BACK checkpoint. For each file you are recovering, ROLL BACK processing determines the start points for rolling the file forward.

The start point is normally the last checkpoint. However, if a discontinuity is detected for the file between the last checkpoint and the end of CCARF, the start point is just after the last discontinuity record.

2. Maintains transaction and constraints logs for each transaction as it is

reapplied.

3. Builds a Transaction Control Block for each concurrent transaction in the original run to keep track of each transaction back out and constraints log. This allows ROLL FORWARD processing to back out transactions that were backed out in the original run.
4. Issues messages indicating what actions are taken for all recovered files, as the CCARF data set is processed.
5. Reapplies file updates from the starting point in each file.
6. Closes recovered files.
7. Closes CCARF.
8. Takes a new checkpoint and allows new updates to begin.
9. Issues the following message:

```
*** RECOVERY IS NOW COMPLETE ***
```

ROLL FORWARD file types

For system recovery purposes, Model 204 files can be categorized according to the processing algorithm applied to determine the stopping point of the ROLL FORWARD processing. Each file type is handled differently during the roll forward phase of recovery. The ROLL FORWARD algorithm is determined by the specified file recovery option, FRCVOPT:

Roll-forward-all-the-way files - FRCVOPT=X'09'

Recovery of roll-forward-all-the-way files ignores update unit boundaries. For these files, ROLL FORWARD processing applies all individual updates that have been logged to CCAJRNL. This minimizes the loss of work during ROLL FORWARD processing by sacrificing logical consistency.

Transaction back out files FRCVOPT X'08' bit off

For transaction back out files, ROLL FORWARD processing initially reapplies all individual updates that have been logged to the journal, just as it does for roll-forward-all-the-way files. When the end of the journal is reached, any incomplete transactions are backed out.

Because the lock pending updates mechanism acquires an exclusive enqueue on records updated by a transaction until the transaction either commits or is backed out, concurrent transactions running against the same files are logically independent of one another.

Journal data set in ROLL FORWARD logging

The journal data set is used as the log for ROLL FORWARD processing. If ROLL FORWARD logging is active for a file (based on the values of the RCVOPT system parameter and the FRCVOPT file parameter), each change made to the file is logged at the time it is made. In addition to the change entries, important events such as checkpoints and discontinuities are also logged on the journal. Along with update unit boundaries, detailed in Chapter 19, these special markers are used to determine where the process of rolling forward should begin and end for each file.

The journal buffer is written out to disk or tape by the completion of every update unit that corresponds to a User Language request or host language program. This minimizes the amount of ROLL FORWARD data that can be lost during a system crash, and ensures that the commit of a transaction is always reflected on disk, in the CCAJRNL data set, at the time of the commit.

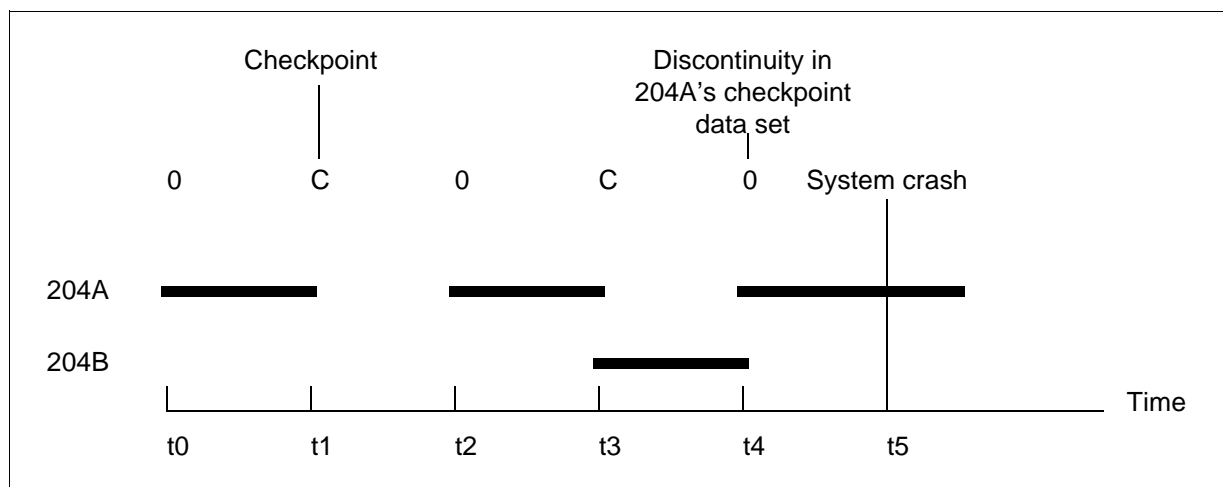
File discontinuities

The ability to share files between two distinct copies of Model 204 can create an operational problem. If correct operating procedures are not followed, the use of checkpoint and roll forward facilities might result in a loss of file integrity under certain circumstances.

For example, suppose that one file is updated by two or more jobs. In Figure 16-1:

- 204A is an Online job that has file A opened for update at points t_0 , t_2 , and t_4 on the chart.
- 204B is a batch job submitted to update file A.

Figure 16-1. Checkpoint data set discontinuity



Suppose that a system crash occurs at t_5 and the most recent checkpoint occurred at t_1 . File A can be rolled back to t_4 ; at this point, physical consistency

is guaranteed, because the file was just opened. File A cannot be rolled back any further than t_4 , because older preimages might partially, but not completely, undo updates from 204B. There is a gap, or *discontinuity*, in 204A's checkpoint data set that prevents rolling back file A beyond t_4 .

Resolving discontinuities

To resolve the problem presented by this discontinuity, Model 204 writes a discontinuity record on the checkpoint data set when OPEN processing recognizes that another job has updated the file. Model 204 does not roll back changes to a file across a discontinuity record.

In the Figure 16-Figure 16-1., if a crash occurs at t_5 , Model 204 restarts by processing job 204A's checkpoint data set from beginning to end and then reverses to restore preimages. On this reverse pass, file A is restored to its state at t_4 ; the discontinuity record stops file A from being rolled back to the checkpoint at t_1 . All other files are rolled back to the checkpoint at t_1 .

Resolving logical inconsistencies

When files are rolled back to different points, the possibility exists for logical inconsistencies to appear in the database. Model 204 provides an option that prevents a second job (204B in the example) from updating a file that already has been updated by another job that is still running (204A). Through enqueueing, 204A retains shared control of file A even when file A is closed. 204B can open file A for retrievals but not updates. Use the FRCVOPT parameter to indicate whether the file is held or released when it is closed.

How discontinuities occur

Besides the two-job update case just described, a discontinuity is produced by any operation that does not write a complete set of preimages for the pages it updates. Several commands, including CREATE, INITIALIZE, REGENERATE, and RESTORE, cause discontinuities. Some of these commands rewrite the entire file, making preimages impractical, or, in the case of CREATE, impossible. The REGENERATE command logs a discontinuity record, because roll forward logging is automatically turned off during media recovery.

Understanding update units

An **update unit** is any sequence of operations that is allowed to update one or more files. See Chapter 19 for a complete description of update units. Update units include update operations that can be backed out (back out update units or transactions) and update operations that cannot be backed out. Each update unit is assigned a unique ID by the system. This update ID is written to the audit trail when the update unit starts and ends. See "Reporting facilities" on page 367 for the text of the messages issued to the audit trail at these times.

Update unit boundaries

The boundaries of a complete update unit depend on a number factors: whether or not the update unit can be backed out, the updating commands or statements involved, the kinds of operations and/or situations involved in the update, and special events like hard restarts or request cancellations. A detailed listing of the beginnings and ends of update units is found in Chapter 19.

Determining the starting point for ROLL FORWARD processing

Normally, ROLL FORWARD processing begins reapplying updates at the checkpoint selected by ROLL BACK processing. However, if a discontinuity occurred for a file between the last checkpoint and the time of the crash (see “File discontinuities” on page 362), the file cannot be rolled back to the checkpoint. ROLL FORWARD processing cannot reapply any updates to the file between the checkpoint and the discontinuity, although updates that occurred after the discontinuity can be reapplied. Simply stated, ROLL FORWARD processing starts where ROLL BACK processing stopped for each file.

ROLL FORWARD processing and user hard restarts

Restarts that occur during active transactions cause a request cancellation and can optionally cause a back out. A user hard restart occurs when a serious error is encountered during the processing of a user’s request or command. Such errors include physical I/O errors in a file or physical inconsistencies in the database. The user is logged out and all files are closed. Those files are marked physically inconsistent. They cannot be used again until you clear the condition.

Repairing physically inconsistent files

Most of the methods for fixing a physically inconsistent, broken file, such as using the INITIALIZE command or running a batch job, produce a discontinuity. One method, resetting the physically inconsistent flag in FISTAT, actually does not fix the file but does make it available for use, presumably so that its contents can be dumped in some form for reloading. To extract the contents of a broken file successfully, you might need to make further changes to the file to avoid the broken areas.

Update units and hard restarts

A hard restart normally leaves an update unit that cannot be backed out incomplete. (Restart during a transaction can optionally cause the transaction to be backed out.) If the system crashes after a hard restart but before RESTART recovery is initiated, the restarted update unit is used as the ROLL FORWARD stopping point for the files involved. If one or more of the files is cleared by a discontinuity before the system crash, that file is no longer part of

the restarted update unit. Nothing is reapplied to the file until after the discontinuity. For files that are not cleared, the restarted update unit remains incomplete.

Back outs during ROLL FORWARD processing

For transaction back out files, a transaction can be backed out during ROLL FORWARD processing for two reasons:

- Transactions that are backed out in the original run are also backed out during ROLL FORWARD processing. In the original run, individual updates are logged to the journal as they occur. If the transaction is backed out instead of committed, this is reflected in the journal. During ROLL FORWARD processing, the individual updates are reapplied based on the journal entries logged for them by the original run. If ROLL FORWARD processing encounters a back out record on the journal, it proceeds to back out all the individual updates it reapplied since the transaction began.
- Transactions that are still incomplete (that is, uncommitted at the time of the crash) when the end of the journal is reached are automatically backed out by ROLL FORWARD processing.

Logging ROLL FORWARD processing

To accomplish transaction back outs during ROLL FORWARD processing, Model 204 maintains transaction back out and constraint logs for each active transaction during Pass 2 of ROLL FORWARD processing. ROLL FORWARD transaction back out and constraint logs are built in CCATEMP, just as they were in the original run. Thus, ROLL FORWARD processing requires as much CCATEMP space for transaction back out and constraint logging as the original run. Insufficient CCATEMP space terminates recovery.

In addition, ROLL FORWARD processing builds a special transaction control block for each possible concurrent transaction in the original run to keep track of each transaction's transaction back out and constraint log. This requires that additional storage be allocated during Pass 2 of ROLL FORWARD processing. Add this storage to SPCORE for recovery runs. The formula for calculating this additional storage is:

$$16 + NUSERS * 48 + (NFILES/8 \text{ rounded up to the nearest multiple of } 4) = \text{bytes of additional storage required}$$

where:

NUSERS and *NFILES* are taken from the original run.

For example, recovery of a Model 204 run with *NUSERS*=100 and *NFILES*=50 requires allocation of the following additional bytes of storage during ROLL FORWARD, Pass 2:

$$16 + 100(48 + 50/8 \text{ rounded up to the nearest multiple of } 4)$$

4) = 5616

RESTART recovery considerations for sub-transaction checkpoints

If you are recovering a job that was using sub-transaction checkpoints, the RESTART process is going to need all the checkpoint information from that job. Therefore, in addition to the RESTART data set definition for the CHKPOINT data set from the failed job, you must add a RESTARTS data set definition that points to the CHPNTS data set from the failed job.

If you run RESTART recovery as a single-user job step ahead of your Online, do not add CPTYPE=1. The recognition of a RESTARTS data set is the indicator that the CPTYPE=1 job is being recovered.

Considering Roll Back and Roll Forward

- ROLL BACK
When RESTART rolls back to a sub-transaction checkpoint it then backs out any updates that had not committed when the sub-transaction checkpoint was taken.
- ROLL FORWARD
The roll forward process is the same as rolling forward from a transaction checkpoint.
- Running ROLL BACK and ROLL FORWARD as separate steps.
You can run a RESTART ROLL BACK and subsequently request just a RESTART ROLL FORWARD, however, in the case of sub-transaction checkpoint the ROLL BACK process will always occur even if you previously rolled back.

For RESTART ROLL BACK only processing

RESTART ROLL BACK processing requires RCVOPT=8, even if ROLL FORWARD is not specified. Roll back only to a sub-transaction checkpoint is available, so this type of recovery requires journaling of information for subsequent secondary recovery, if needed. In addition, GDGs are supported for CHKPOINT during recovery, which allows for unlimited size recovery checkpoints.

If at CHKPOINT open during RESTART ROLL BACK processing there is no CHKPOINT defined for the job, the job displays the following message and waits for a response from the operator.

```
OO ///  UNABLE TO OPEN CHKPOINT,  REPLY RETRY OR CANCEL
```


Restarting after a system failure

To restart Model 204 or an IFAM4 application after a system failure issue a RESTART command that is the first command immediately following the last user (IODEV) parameter line in the CCAIN input stream. The ROLL BACK and ROLL FORWARD arguments activate the ROLL BACK and ROLL FORWARD facilities used for file recovery.

The following considerations apply to using the RESTART command:

- The checkpoint ID of the TO clause in the RESTART command must correspond exactly to the checkpoint ID displayed in a `CHECKPOINT COMPLETED` message from the Model 204 run being recovered. The option to specify the checkpoint is useful in an environment in which checkpoints are coordinated with Host Language Interface program checkpoints.

- Files broken as a consequence of a user hard restart can be fixed by rolling back to a checkpoint prior to the current RESTART command.

If the ROLL FORWARD option is also requested, the recovered file contains all the updates completed before the user restart. This method of fixing files requires running RESTART recovery after the end of the run that broke the files and before the files are updated by any other run.

- To run the ROLL FORWARD facility, sufficient space must be available in spare core, SPCORE, for special control blocks that are allocated when ROLL FORWARD processing begins and freed when it is completed.

One other block is allocated, with a size given by:

$$16 + (36 * \text{NUSERS})$$

where the value of NUSERS is from the run that crashed, not from the recovery run.

Reporting recovery status

The final status of each file that participated in recovery is reported at the conclusion of recovery processing, when you issue the STATUS command. The effects of the most recent recovery on an individual file are reported when the file is opened with update privileges.

Reporting facilities

Update unit ID numbers and audit trail messages help you determine exactly which updates were being made when a system crash occurred and which updates were not reapplied by RESTART recovery.

UPDTID user parameter

The view-only, user parameter, UPDTID, lets applications determine the current update unit number. The update unit number is a sequential number

that provides a unique identifier for each update unit in a Model 204 run. The update ID is incremented for each update unit. When an update unit is active, UPDTID contains the update ID, otherwise it contains zero. Applications can use this parameter to keep track of the most recent update unit started by each user.

Audit trail messages

At the beginning and end of each update unit, The following messages are written to the audit trail:

```
*** M204.0173: START OF UPDATE n AT hh:mm:ss.th
```

and:

```
*** M204.0172: END OF UPDATE n AT hh:mm:ss.th
```

Message M204.0173 is issued for every update unit. The default destination of both messages is the audit trail. The system manager can use the MSGCTL command to change the destination of these messages.

Reporting ROLL FORWARD processing

Model 204 lists each update unit that ROLL FORWARD processing automatically backed out or partially reapplied, because the update unit was incomplete when the end of the journal was reached. The message is in the following format:

```
*** M204.1200: ROLL FORWARD {BACKED OUT | PARTIALLY REAP-  
PLIED} UPDATE n FOR THE FOLLOWING FILES:
```

where:

n is the update ID.

A second message lists the files updated by the update unit:

```
*** M204.1214: filename
```

Final status of files affected by RESTART recovery

For files that are rolled back and not rolled forward a message in the following format is issued:

```
*** M204.0621: ROLLED BACK TO {CHECKPOINT | DISCONTINUITY}  
OF day month
```

For each file rolled forward, either or both of the following messages is issued:

```
*** M204.0622: UPDATE n OF day month year hh:mm:ss.th WAS  
HIGHEST UPDATE FULLY REAPPLIED TO filename BY [ROLL FOR-  
WARD | REGENERATE]
```

```
*** M204.0622: UPDATE n OF day month year hh:mm:ss.th WAS
LOWEST UPDATE {BACKED OUT | PARTIALLY REAPPLIED} TO file-
name BY [ROLL FORWARD | REGENERATE]
```

where:

n is the update ID.

Note: For update units backed out or partially reapplied, only the lowest update ID is reported. A complete list of such update units for a particular file can be obtained from the message M204.1200 described above. Those messages are available only in the recovery run, CCAUDIT.

Opening files and their status reports

At the end of recovery, the X'10' bit of the FISTAT parameter is set to indicate that the file was recovered. Subsequently, when you open a file, messages report the effects of the most recent RESTART recovery, until you reset FISTAT to X'00'. The status reported during open processing is as follows:

```
*** M204.1203: filename WAS LAST UPDATED ON day month year
hh:mm:ss.th
```

```
*** M204.1238: RECOVERY OF filename WAS LAST REQUIRED ON
day month year hh:mm:ss.th
```

```
*** M204.0621: ROLL BACK TO {CHECKPOINT | DISCONTINUITY}
OF day month year hh:mm:ss.th
```

or:

```
*** M204.0622: UPDATE n OF day month year hh:mm:ss.th WAS
{HIGHEST | LOWEST} UPDATE {FULLY REAPPLIED | BACKED OUT |
PARTIALLY REAPPLIED} TO filename BY ROLL FORWARD
```

Suppressing status reporting

You can suppress recovery status messages when files are opened by resetting the X '10' bit of the FISTAT file parameter to zero with the RESET command.

Requesting status reports

Use the STATUS command to obtain reports on the recovery status of a file in the same job that executes the RESTART or REGENERATE command.

The command is issued in the following form and applies to all operating systems:

```
STATUS filename
```

where *filename* identifies a file opened during the current run by a user or a RESTART command.

If a user with system manager or system administrator privileges issues the STATUS command, you can omit the file name to list status information for all files opened during the run.

One or two lines of information are listed for each file, in the form:

```
filename: current-status
RESTART-status
```

where:

- *current-status* indicates special conditions that can affect discontinuities and future recovery of the file.

The values of current status are CLOSED, DEFERRED, ENQUEUED, OPEN, and UPDATED; see the *Rocket Model 204 Parameter and Command Reference*.
- *RESTART-status* indicates a file that participated in recovery in the current run.

The values of *RESTART status* are listed in Table 16-1

Table 16-1. RESTART command status values

Status	Description
BEING RECOVERED	The file is in the recovery process when the STATUS command is issued.
NOT RECOVERED	<p>The file cannot be opened until the RESTART command ends. One of the following reasons is given:</p> <ul style="list-style-type: none"> • DEFERRED UPDATE CHKP MISSING The scan of the deferred update stream did not locate the checkpoint used for ROLL BACK processing • DEFERRED UPDATE READ ERROR The scan of the deferred update stream not completed because of a read error. • DEFERRED UPDATES MISSING The deferred update stream could not be opened. • MISSING The file could not be opened. • OBSOLETE The file was updated by a Model 204 job after the last update in the run being recovered. • IS ON A READ-ONLY DEVICE The file is on a device for which Model 204 has read-only access.
ROLLED BACK TO CHECKPOINT...	The file was rolled back to the checkpoint taken at the given date and time. The file was not rolled forward.

Table 16-1. RESTART command status values (Continued)

Status	Description
ROLLED BACK TO DISCONTINUITY...	The file was rolled back to the discontinuity occurring at the given date and time. The file was not rolled forward.
ROLLED FORWARD TO UPDATE...	The file was rolled forward. The last update reapplied by roll forward had the specified ID and originally ended at the given date and time.
REGENERATED TO CHECKPOINT...	The file was recovered using media recovery (REGENERATE). Updates were reapplied up to the time of the checkpoint given.
REGENERATED TO UPDATE...	The file was recovered using media recovery (REGENERATE). Updates were reapplied up to and including the specified update.

Recovery data sets and job control

Data sets required for RESTART recovery

The data sets required for z/OS, z/VM, and z/VSE for RESTART recovery are listed in Table 16-2.

Table 16-2. RESTART recovery data sets

Model 204 file name	Required...
CCAJRNL	<p>(Recommended) When running RESTART recovery as a separate job step, CCAJRNL is not required, unless FRCVOPT=8.</p> <p>(Not recommended) When running RESTART recovery in the same step that brings up the Online, CCAJRNL is required for logging all updates (RCVOPT=8) that will occur in the Online.</p>
CCARF	When restarting the system. For ROLL FORWARD processing, the CCARF stream must be the same as the CCAJRNL stream from the run being recovered.
CHKPOINT	For taking checkpoints (RCVOPT=1) and for logging preimages of updated file pages.
RESTART	<p>When restarting the system. For ROLL BACK processing, the RESTART stream must be the same as the CHKPOINT stream of the run being recovered.</p> <p>(For z/VSE only) The RESTART file is used as a direct access file. DLBL code DA must be specified.</p>

Sample z/OS JCL for a recovery run

The JCL for the original run specifies the following data sets:

```
//CHKPOINT DD DSN=M204.CHKP1,DISP=OLD  
//CCAJRNL DD DSN=M204.JRNL1,DISP=OLD
```

The JCL of the recovery run with roll forward is as follows:

```
//RESTART DD DSN=M204.CHKP1,DISP=OLD  
//CCARF DD DSN=M204.JRNL1,DISP=OLD  
  
//CHKPOINT DD DSN=M204.CHKP2,DISP=OLD  
//CCAJRNL DD DSN=M204.JRNL2,DISP=OLD
```

Note: When performing a RESTART recovery, the RESTART and CCARF streams must be provided as input to an ONLINE or BATCH204 configuration.

The RESTART stream is the CHKPOINT stream from the run being recovered. Although created as a sequential output file, the RESTART stream is treated as a direct access file if on disk.

You must provide a new CHKPOINT stream for checkpoint output generated during the restart process. CCAJRNL is optional in a RESTART-only job step with NUSERS=1.

Sample z/VSE JCL for recovery

The following considerations apply to the required JCL statements in a z/VSE environment:

- If checkpoint and journal data sets are disk resident, they are used as sequential output files that do not require allocation with the ALLOCATE utility.
- You must provide DLBL and EXTENT statements for any recovery or restart streams that are disk resident.
- ASSGN statements might be necessary.
- TLBL statement is required instead of DLBL and EXTENT statements if magnetic tape is substituted for a disk file:
 - The DOS file name specified on the TLBL statement must take the form of a programmer logical unit number (SYSnnn) and must be the same as the programmer logical unit number on the ASSGN statement for the tape drive used as the input or output device.
 - You must specify the programmer logical unit number in the FILENAME parameter of a DEFINE DATASET command for the corresponding Model 204 internal file name.
 - The DEFINE DATASET commands for recovery or restart files on tape must precede the User 0 parameter line in the CCAIN input stream.

Each file must be referred to by its Model 204 internal name.

- When recovering Model 204 with RESTART processing, you must provide the RESTART and CCARF streams as input to an ONLINE or BATCH204 configuration.
- The RESTART stream is a CHPNT stream from the run being recovered. Although created as a sequential output file, the RESTART stream is treated as a direct access file if on disk.
- You must provide a new CHPNT stream for checkpoint output generated during the restart process. CCAJRNL is optional in a RESTART-only job step with NUSERS=1.
- The lack of a DLBL statement with DA specified in the CODES parameter results in an error message from z/VSE during the processing of the RESTART command.
- The following message results from treating the CHPNT stream as a direct access file when it is used as input to the restart process:

```
M204.0139: EOF ASSUMED IN FIRST PASS OF RESTART DATASET
```

You can ignore the message.

z/VSE example 1: checkpoint and journal files on disk (FBA)

The number of blocks and tracks used in this example are arbitrary values.

```
//JOB MODEL204
.
.
.
// DLBL CHPNT, 'MODEL204.CHPPOINT.FILE', 99/365, SD
// EXTENT SYS021, SYSWK1, , 1000, 13000
// ASSGN SYS021, DISK, VOL=SYSWK1, SHR
// DLBL CCAJRNL, 'MODEL204.JOURNAL.FILE.1', 99/365, SD
// EXTENT SYS021, SYSWK1, , 14000, 13000
.
.
.
// EXEC ONLINE, SIZE=AUTO
PAGESZ=6184, RCVOPT=9, ...
.
.
/*
/ &
```

z/VSE example 2: checkpoint and journal files on tape

```
// JOB MODEL204
```

```
.  
.br/>// TLBL SYS041,'M204.CHKPOINT',99/365  
// ASSGN SYS041,TAPE  
// TLBL SYS040,'M204.JOURNAL',99/365  
// ASSGN SYS040,TAPE  
.  
.br/>// EXEC ONLINE,SIZE=AUTO  
DEFINE DATASET CCAJRNL WITH SCOPE=SYSTEM FILENAME=SYS040  
DEFINE DATASET CHPNT WITH SCOPE=SYSTEM FILENAME=SYS041  
PAGESZ=6184,RCVOPT=9,...  
.  
.  
/*  
/&
```

z/VSE example 3: recovery-restart from disk

In the following example, all files are on disk. The checkpoint, journal, and restart streams are on FBA devices. The ROLL FORWARD stream is on a CKD device. The number of blocks and tracks are arbitrary values.

```
// JOB MODEL204 RESTART  
.  
.  
// DLBL RESTART,'MODEL204.CHKPOINT.FILE',,DA  
// EXTENT SYS021,SYSWK1,,,1000,13000  
// ASSGN SYS021,DISK,VOL=SYSWK1,SHR  
// DLBL CHPNT,'MODEL204.CHKPOINT.FILE.2',99/365,SD  
// EXTENT SYS023,SYSWK2,,,2000,13000  
// ASSGN SYS023,DISK,VOL=SYSWK3,SHR  
// DLBL CCAJRNL,'MODEL204.JOURNAL.FILE.3',99/365,SD  
// EXTENT SYS023,SYSWK3,,,13000,13000  
// DLBL CCARF,'MODEL204.JOURNAL.FILE.2',99/365,SD  
// EXTENT SYS022,SYSWK2,,,30,300  
// ASSGN SYS022,DISK,VOL=SYSWK2,SHR  
.  
.  
// EXEC ONLINE,SIZE=AUTO  
PAGESZ=6184,RCVOPT=9,...  
.  
.  
RESTART ROLL BACK ERROR STOP ROLL FORWARD  
/*  
/&  
PAGESZ=6184
```


z/VSE example 4: recovery-restart from tape

The following example illustrates checkpoint and journal streams on disk, and restart and roll forward streams on magnetic tape. Checkpoint and journal streams are on FBA devices.

```
// JOB MODEL204
.
.
// DLBL CHPNT,'MODEL204.CHKPOINT.FILE',99/365/SD
// EXTENT SYS021,SYSWK1,,,1000,13000
// ASSGN SYS021,DISK,VOL=SYSWK1,SHR
// DLBL CCAJRNL,'MODEL204.JOURNAL.FILE.1',99/365,SD
// EXTENT SYS023,SYSWK2,,,11000,13000
// ASSGN SYS023,DISK,VOL=SYSWK3,SHR
// TLBL SYS040,M204.JOURNAL.2'
// ASSGN SYS040,TAPE
// TLBL SYS041,'M204.CHKPOINT'
// ASSGN SYS041,TAPE
.
.
// EXEC ONLINE,SIZE=AUTO
DEFINE DATASET CCARF WITH SCOPE=SYSTEM FILENAME=SYS040
DEFINE DATASET RESTART WITH SCOPE=SYSTEM FILENAME=SYS041
PAGESZ=6184,RCVOPT=9,...
.
.
RESTART ROLL BACK ERROR STOP ROLL FORWARD
/*
/&
```

Note: Model 204 files for which the DEFINE DATASET command was used to relate the Model 204 internal file name to a z/VSE file name can be recovered with the RESTART facility, only if the DEFINE DATASET command precedes the User 0 parameter line in the CCAIN input stream.

Statements required for z/VM recovery

Use the following FILEDEF statements in the Online run for which you want to provide recovery:

```
FILEDEF CHPPOINT mode DSN M204 chkp1
FILEDEF CCAJRNL mode DSN M204 jrn11
```

The following FILEDEF statements are required for a recovery run requesting ROLL BACK and ROLL FORWARD processing:

```
FILEDEF RESTART mode DSN m204 chkp1
```

```
FILEDEF CCARF mode DSN m204 jrn11
FILEDEF CHKPOINT mode DSN m204 chkp2
FILEDEF CCAJRNL mode DSN m204 jrn12
```

The following considerations apply:

- Do not store CCAJRNL and CHKPOINT on CMS minidisks. If the service machine crashes, CMS data sets are lost.
- Because CMS does not support read-backward from tape, you cannot use a tape checkpoint data set as the restart data set during recovery.
- Checkpoint and journal data set sequencing in the ONLINE EXEC can allow for multiple generations of recovery data sets. Modify FILEDEFs for journal and checkpoint. Add FILEDEFs for the old journal and checkpoint, as shown in the following example.

z/VM recovery examples

- The statements below first determine the last set of CHKPOINT and CCAJRNL files used and then increment the next set by 1.

```
LISTFILE RESTART SEQ* A (EXEC
EXEC CMS &STACK
&READ VARS &FN &FT &FM
&SEQ=&&SUBSTR &FT 4
```

- The following statements calculate the next sequence number, assuming five sets of recovery data sets:

```
&SEQX=&SEQ
&SEQ =&SEQ+1
&IF &SEQ GT 5 &SEQ=1
```

- This statement creates a recovery control data set from the service machine:

```
XEDIT RESTART SEQ0
```

- Insert this line and file the data set:

```
*** DO NOT DELETE THIS DATA SET, IT IS ESSENTIAL TO
MODEL 204 RECOVERY
```

- Rename the control file for the next run and enter the FILEDEFs, as follows:

```
RENAME &FN &FT &FM = SEQ &SEQ
FILEDEF CCAJRNL M DSN M204 CCAJRNL SEQ&SEQ
FILEDEF CCARF M DSN M204 CCAJRNL SEQ&SEQX
FILEDEF CHKPOINT M DSN M204 CHKPOINT SEQ&SEQ
FILEDEF RESTART M DSN M204 CHKPOINT SEQ&SEQX
```

Handling recovery failures

Recovering from failure during recovery is possible only if the cause of the original failure is corrected. A serious error encountered during RESTART recovery results in a message describing the error, followed by a message indicating that the run is aborted.

Determining the cause of the error

When a recovery run aborts, the cause of the error can be determined by examining the job step return code and messages written to CCAUDIT:

- If the RESTART command does not complete normally and the Model 204 run stays up, the return code is 52.
- If the Model 204 run is terminated during evaluation of RESTART recovery, the job abends with a code of 999.

Once the cause of the error is eliminated, the recovery job can be run.

Correcting errors

Use the following strategies to correct errors:

- If the error is an I/O error on either the RESTART or CCARF recovery streams, use IEBGENER or a similar sequential copy utility to copy the affected stream to another volume. Then rerun recovery using the copy.
- If the problem cannot be corrected, but involves only one file, remove the file from the recovery run and recover it separately.

Automated secondary recovery: reuse JCL

You do not need to alter your JCL when you run RESTART recovery multiple times. Occasionally, when RESTART processing is underway, it might be interrupted:

- During ROLL BACK processing
- During ROLL FORWARD processing

If the interruption occurs:

- During ROLL BACK processing, you resubmit the job.
- During ROLL FORWARD processing, you resubmit the job. Make *no* changes to the RESTART or CHKPOINT data sets.

Model 204 determines whether this is a primary or secondary recovery run and uses the appropriate file: RESTART or CHKPOINT for ROLL BACK recovery.

Note special requirement for z/VSE: To take advantage of the automation of secondary recovery, you must include CHPNTD in your primary recovery job. CHPNTD uses the same data set name as CHPNT, but specifies DA rather than SD. Specify CCAJRN, CCARF and RESTART as always. For example:

```
// DLBL CHPNTD,MODEL204.CHKPOINT.FILE.2,0,DA
// EXTENT SYS023,SYSWK2,,,2000,13000
// DLBL CHPNT,MODEL204.CHKPOINT.FILE.2,0,SD
// EXTENT SYS023,SYSWK2,,,2000,13000
// ASSGN SYS023,DISK,VOL=SYSWK3,SHR
```

If you do not include CHPNTD, then you must run secondary recovery.

CHKPNTD data set under z/VSE

When a CHPNTD data set is defined to take advantage of the automation of secondary recovery, Model 204 can verify whether or not the CHPNT data set is large enough for ROLL FORWARD processing.

If the CHPNT data set is too small, Model 204 issues the following error message and terminates before any actual recovery processing begins:

```
M204.2605: CHKPOINT TOO SMALL FOR ROLL FORWARD - number1
BLOCKS REQUIRED; number2 FOUND
```

Sizing the checkpoint data sets correctly

For z/OS sites, Model 204 verifies that the new CHKPOINT data set is large enough to contain all the information that needs to be written to the CHKPOINT file during ROLL FORWARD processing. If the new CHKPOINT data set is too small, Model 204 issues the following error message and terminates before any actual recovery processing begins:

```
M204.2605: CHKPOINT TOO SMALL FOR ROLL FORWARD - number1
BLOCKS REQUIRED; number2 FOUND
```

The CHKPOINT data set must also be a single-volume data set due to BSAM limitations regarding read-backwards.

If you use sub-transaction checkpoints, as well as transaction checkpoints, define the CHPNTS data set to match the CHKPOINT data set.

Rerunning RESTART recovery after a successful recovery

If, for any reason, you want to force a rerun of your successful recovery job, you must define a new CHKPOINT data set. If you rerun your recovery job without a new CHKPOINT data set, recovery will be bypassed and the following message is displayed:

```
M204.0143: NO FILES CHANGED AFTER LAST CP, RESTART BYPASSED
```

ROLL BACK processing

The ROLL BACK process ends with a checkpoint.

When ROLL BACK processing is completed, the following messages are displayed:

```
M204.0158: END OF ROLLBACK
M204.0843: CHECKPOINT COMPLETED ON yy.ddd hh:mmss.th
```

CHKPOINT data set required

Changing the point at which the checkpoint is taken has the following impact. The CHKPOINT data set is required for any RESTART recovery processing, because the ROLL BACK facility needs the CHKPOINT data set to write the end-of-processing ROLL BACK checkpoint.

If you do not define a CHKPOINT data set, the following message is issued:

```
M204.1300: RESTART COMMAND REQUIRES CHECKPOINT LOGGING -
RUN TERMINATED
```

Operational changes to ROLL FORWARD processing

Tracking the application of updates

As the CCARF data set is processed by ROLL FORWARD, the following messages are printed each time a new journal block is read in which the hour in the date-time stamp at the beginning of the journal block is one hour (or more) greater than the hour in the last M204.1992 message printed. For example,

```
M204.1992: RECOVERY: PROCESSING ROLL FORWARD BLOCK#
0000001B 01.235 16:31:32.43
M204.1992: RECOVERY: PROCESSING ROLL FORWARD BLOCK#
00001C57 01.235 17:00:23.34
M204.1992: RECOVERY: PROCESSING ROLL FORWARD BLOCK#
00002F05 01.235 18:00:12.38
```

These messages are intended to provide an indication that ROLL FORWARD processing is progressing and help you estimate when recovery will complete. The ROLL FORWARD BLOCK# is the sequential number (from the beginning of CCARF) of the journal block read at that point during ROLL FORWARD processing.

ROLL FORWARD processing can be run separately

ROLL FORWARD processing can be invoked subsequent to a successful ROLL BACK process in a separate job or job step by issuing the following command:

```
RESTART ROLL FORWARD
```

The following message does not indicate that the full ROLL BACK processing is occurring; this message indicates that the correct starting point for ROLL FORWARD processing is being located.

```
M204.2512: ROLL BACK WILL USE THE FOLLOWING dataset:  
RESTART | CHKPOINT
```

If Model 204 detects that a successful ROLL BACK process did not occur, it forces a full ROLL BACK process and all the standard ROLL BACK messages are displayed.

Setting RCVOPT when running ROLL BACK and ROLL FORWARD separately

When running ROLL BACK-only, Rocket Software recommends setting RCVOPT parameter to 0. If you then run a subsequent ROLL FORWARD-only step, the ROLL BACK processing will be suppressed. If you do run ROLL BACK-only with RCVOPT set including 1, do not set CPMAX TO 0 OR 1.

- If you run ROLL BACK-only with an RCVOPT setting that does include X'01' and the CPMAX parameter is set to 0 or 1, a subsequent ROLL FORWARD-only request must process ROLL BACK again to find the correct checkpoint for ROLL FORWARD processing to locate and apply the ROLL FORWARD updates.
- If in your ROLL BACK-only step RCVOPT included X'01' and the CPMAX parameter was set greater than one, you can suppress the ROLL BACK processing on a ROLL FORWARD-only step by supplying the ROLL BACK TO checkpoint ID information that was previously rolled back to. Issue a RESTART command, as shown in the following syntax:

```
RESTART ROLL BACK TO yy.ddd hh:mm:ss.th ROLL FORWARD
```

ROLL FORWARD-only processing uses this information to locate the correct ROLL FORWARD starting point in the CCARF data set.

If you do not provide the checkpoint ID information, then the ROLL BACK processing is automatically triggered to locate the correct starting point for ROLL FORWARD processing.

- Under the previously noted conditions, if you do not provide the checkpoint ID information, then the ROLL BACK processing is automatically triggered to locate the correct starting point for ROLL FORWARD processing.
- Also, if running a ROLL BACK-only process, the CCARF file definition is no longer required.

ROLL FORWARD processing, in any context, requires RCVOPT=1. CPMAX should be set to a value that meets your site's need. Your current RESTART configuration should reflect whether you take the CPMAX default value or set some other value.

Finding checkpoint in journal

Locating the checkpoint record rolled back to in ROLL BACK processing is done using the NOTE/POINT facility provided for BSAM data sets. The NOTE/POINT facility allows direct access to a single record in a file of any size. This eliminates the potentially long delay currently encountered between the end of ROLL BACK processing and the beginning of ROLL FORWARD processing. The NOTE/POINT facility handles all forms of CCARF data sets, including ring streams, parallel streams, and single sequential data sets.

Note special requirement for z/VSE: IBM restricts the NOTE/POINT facility for sequential data files under z/VSE.

RESTART recovery requirements - CCATEMP, NFILES, NDIR

Checkpoint information that is logged can include data for RESTART recovery that must be relocated back to the same CCATEMP pages and using the same internal file numbers during RESTART recovery that were used during the Online run. Therefore, in a RESTART job or step the CCATEMP size, NDIR, and NFILES must be equal to or greater than the values after initialization of the job being recovered.

The RESTART value for:

- NFILES may need increase by as much as two greater than set in the Online CCAIN, if CCAGRP and CCASYS were opened by the Online.
- NDIR may need to increase by as much as three to accommodate CCATEMP, CCAGRP, and CCASYS, if they were opened by the Online.
- CCATEMP in RESTART must be at least as large as that in the job that is being recovered. Verify that the CCATEMP data set allocation or TEMPPAGE in RESTART meets this requirement.

If not properly set for RESTART, you will receive the following error message that specifies the required value and initialization is terminated. Reset the parameter to the value indicated in the message and resubmit the RESTART job.

```
M204.0144 parameter=value BUT MUST BE AT LEAST value (reason)
```

Recovering deferred update mode files

Under the z/OS and z/VM operating systems, files that were in deferred update mode at the time of a crash can be recovered using the ROLL BACK facility either alone or in conjunction with the ROLL FORWARD facility. Checkpoint markers are written on every open, deferred update data set.

How recovery works for deferred update files

For each file being recovered, ROLL BACK processing opens the file's deferred update data set(s) for input. Each deferred update data set is scanned for a marker that corresponds to the last checkpoint. The data set is then changed from input to output, so that changes made during ROLL FORWARD processing are written over changes that were backed out. At the end of recovery, files in deferred update mode are not closed. Changes made during the new run are added to the old deferred update data set.

If ROLL BACK processing cannot open the file's deferred update data set, or if the deferred data set or data sets do not contain the correct checkpoint marker, the file is not recovered.

Deferred update recovery not supported under z/VSE

Support for recovery of files in deferred update mode is not implemented under the z/VSE operating system.

Recovering dynamically allocated data sets

The RESTART command, by default, dynamically reallocates and recovers any Model 204 files or deferred update data sets that were dynamically allocated in the run that is being recovered.

How recovery works for dynamically allocated data sets

Dynamically allocated data sets normally are closed and freed after recovery processing. Therefore, it is necessary to reallocate the data sets after recovery completes. However, if a dynamically allocated file were opened in deferred update mode in the run that is being recovered, the file in deferred update mode remains allocated and open after recovery completes. Any deferred update data sets that were dynamically allocated also remain allocated and open. This ensures that any further deferred updates after recovery do not overwrite previous records in the deferred update data set.

Bypassing dynamic file allocation

If RESTART recovery processing determines that no files need to be recovered, dynamic allocation is bypassed, which saves CPU and wall clock time.

Media recovery

You can perform media recovery using the REGENERATE or REGENERATE ONEPASS command in a Batch204 job or a single-user Online.

Media recovery procedures are useful for restoring a file when:

- Hardware error occurs on the storage media

- Data set is accidentally deleted
- ROLL FORWARD processing fails for an individual file

Media recovery can automatically restore from a dump of the file, or can allow the restore to be done using any desired technique prior to running media recovery.

Media recovery is initiated by the REGENERATE or REGENERATE ONEPASS command in the CCAIN input stream. Processing reapplies updates made to single or multiple files since the time of a backup of those files.

File regeneration cannot occur across discontinuities or across Model 204 release boundaries.

Phases of a media recovery run

A media recovery run consists of the following phases:

1. Phase 1 — The REGENERATE or REGENERATE ONEPASS command is parsed for syntax errors. If an error is encountered, the command is rejected and no processing is performed.
2. Phase 2 — All files for which dump files are specified in the REGENERATE or REGENERATE ONEPASS command are restored. This step is performed only if media recovery is performing the restore.
3. Phase 3 — The input journal file is processed in one or two passes.

Input journal processing for the REGENERATE command

The last phase of media recovery is performed in two passes as follows:

- Pass 1 — The input journal is scanned for the starting and stopping point of all participating files:
 - If any errors are detected, processing terminates for the files affected by the error.
 - If any files remain to be processed after Pass 1 is completed, the input journal is closed and reopened.
 - If the REGENERATE command BEFORE clause is used, the last complete update or checkpoint taken before the time specified is determined.
- Pass 2 — The input journal is reread and updates that were originally made between the starting and stopping points are reapplied to the files.

Input journal processing for the REGENERATE ONEPASS command

As the journal records are read the updates are applied:

- The input journal is scanned for the starting and stopping point of all participating files. If any errors are detected, processing terminates for the files affected by the error.
- The input journal is read and updates that were originally made between the starting and stopping points are reapplied to the files.

If ONEPASS is specified for a REGENERATE command and the file being regenerated has been recovered within the supplied journals, the regeneration fails with the following error:

```
M204.2629: ONEPASS DISALLOWED ACROSS FILE RECOVERY
```

The file is deactivated and contains all updates as of the time of the recovery. If regeneration is desired across a recovery, two passes of the journal are required.

REGENERATE processing has no back out capability. In ONEPASS each update is applied as read from the journal. This causes a problem if a file is rolled back. The way two passes works is that it reads the entire journal and then sets starting and stopping points for the second pass. This way the updates that are unneeded can be omitted as REGENERATE processing reads past them.

Running REGENERATE with the IGNORE argument

Using the new IGNORE argument with a REGENERATE command bypasses the file parameter list (FPL) update timestamps. This allows you to run REGENERATE processing with one CCAGEN at a time, instead of requiring a single concatenated journal. If necessary, you can run single CCAGEN data sets one at a time in multiple REGENERATE steps.

For example, the first REGENERATE command may be as follows and provide the first journal:

```
REGENERATE FILE abc FROM dumpabc
```

A subsequent run may provide the second journal and specify:

```
REGENERATE FILE abc IGNORE
```

The second run picks up from where the first REGENERATE processing ended and applies the second journal updates.

Previously, you had to concatenate all journals into one data set or specify using a concatenated CCAGEN DD statement.

Caution: If you omit a journal, it is not reported. Therefore, use the new option with care.

The IGNORE keyword is not valid with the FROM option. See *Model 204 Parameter and Command Reference* for syntax details.

Number of files that can be regenerated

The number of files that can be regenerated in a single run depends on the settings of the following parameters on User 0's parameter line:

- NDCBS, the number of data sets associated with the files being regenerated.
- NDIR, the number of file directory entries, must be at least as large as the number of files to be regenerated in a single media recovery run.
- NFILES, the number of file save areas to allocate, must be at least as large as the number of files to be regenerated in a single media recovery run.

Additional files might be needed if the run performs functions in addition to the REGENERATE operation.

Using the REGENERATE or the REGENERATE ONEPASS command

Choosing to use REGENERATE or REGENERATE ONEPASS

If you must regenerate across a recovery process, two passes of the journal are required. You must build REGENERATE commands. In this case, a REGENERATE ONEPASS command will terminate with an error.

Specifying a starting point

The starting point for reapplying updates to a file is the first start of an update unit with a time greater than the last updated time of the file being processed.

- You can request a specific or nonspecific stopping point as an option of the REGENERATE command.
- You can request only a specific stopping point as an option of the REGENERATE ONEPASS command.

For syntax and description of the REGENERATE and REGENERATE ONEPASS command, see the *Rocket Model 204 Parameter and Parameter and Command Reference*.

Specifying a stopping point

To request a specific stopping point, use the TO clause of the REGENERATE or REGENERATE ONEPASS command. If you do not indicate a stopping point, all updates in the input journal are applied.

Not specifying a stopping point

To request a nonspecific stopping point, use the BEFORE clause of the REGENERATE command. The BEFORE clause uses the last complete update or checkpoint that occurred before the specified time as the stopping point for

reapplying file updates. You cannot use a BEFORE clause with a REGENERATE ONEPASS command.

Using a FROM clause

When a REGENERATE or REGENERATE ONEPASS command is issued without a FROM clause, the command assumes that the file was previously restored. The following validity checking occurs:

- Physical consistency — If a file is marked physically inconsistent, processing is discontinued.
- Release boundary — If a file was created prior to Model 204, Release 9.0, processing is discontinued.

The following error messages pertain to the REGENERATE command FROM option:

```
*** M204.1711: 'FROM' CLAUSE REQUIRED FOR FILES CREATED  
PRE R9
```

```
*** M204.1426: INVALID 'FROM' CLAUSE
```

```
*** M204.1708: REGENERATE DID NOT PERFORM RESTORE
```

Required data sets for media recovery

The STEPLIB, CCAIN, and CCAPRINT data sets are required for media recovery.

DB1 through DB n , where n is a number corresponding to the last file to be recovered, define the file data sets to be recovered.

If an INCREASE DATASETS command is issued during the recovered time interval, data sets added to the file by the INCREASE also must be specified.

DUMPDB1 through DUMPDB n , where n is a number corresponding to the last file to be recovered, define the dumped versions of the file data sets to be recovered.

The name specified for the file to be regenerated must match the name of the dump file.

If the run completes successfully, Model 204 displays the message:

```
*** M204.1437: REGENERATE IS NOW COMPLETE
```

Possible error and database inconsistency conditions are described in the *Model 204 File Manager's Guide*.

Example: z/OS media recovery run

This example shows the z/OS JCL required to invoke a media recovery run:

```
//REGEN JOB REGEN,MSGLEVEL=(1,1)
//M204 EXEC PGM=BATCH204,PARM='SYSOPT=136'
//STEPLIB DD DSN=LOCAL.M204.LOADLIB,DISP=SHR
//CCAJRNL DD DSN=REGEN.CCAJRNL,DISP=(NEW,CATLG,CATLG),
// VOL=SER=WORK,UNIT=DISK,SPACE=(TRK,(5))
//CCAGEN DD DSN=LOCAL.M204.JOURNAL.840228,DISP=SHR
// DD DSN=LOCAL.M204.JOURNAL.840229,DISP=SHR
// DD DSN=LOCAL.M204.MERGED.JOURNAL.840302,DISP=SHR
//CCAAUDIT DD SYSOUT=A
//CCAPRINT DD SYSOUT=A
//CCASNAP DD SYSOUT=A
//CCATEMP DD DISP=NEW,UNIT=WORK,SPACE
//CCASTAT DD DSN=LOCAL.M204.CCASTAT,DISP=SHR
//CCAGRP DD DSN=LOCAL.M204.CCAGRP,DISP=SHR
//TAPE2 DD DSN=M204.DB1.DEFERF,DISP=SHR
//TAPE3 DD DSN=M204.DB1.DEFERV,DISP=SHR

//DB1 DD DSN=LOCAL.M204.DB1,DISP=SHR
//DB2 DD DSN=LOCAL.M204.DB2,DISP=SHR
//DB3 DD DSN=LOCAL.M204.DB3,DISP=SHR
//DB4 DD DSN=LOCAL.M204.DB4,DISP=SHR
//DUMPDB1 DD DSN=DUMP.DB1,DISP=SHR
//DUMPDB2 DD DSN=DUMP.DB2,DISP=SHR
//DUMPDB3 DD DSN=DUMP.DB3,DISP=SHR
//CCAIN DD
* NFILES=10,NDIR=10,LAUDIT=1,SPCORE=10000
*
* OPEN THE FILE IN DEFERRED UPDATE MODE
* BEFORE THE REGENERATE COMMAND
*

OPEN DB1,TAPE2,TAPE3

REGENERATE
FILE DB1 FROM DUMPDB1
FILE DB2 FROM DUMPDB2 TO LAST CHECKPOINT
FILE DB3 FROM DUMPDB3 TO UPDATE 3 OF 83.216 06:28:42:98
FILE DB4 TO CHECKPOINT 83.216 01:05:22:99
END
EOJ
```

Media recovery NonStop/204

Media recovery restores an individual Model 204 file, not the entire database as discussed in “ROLL BACK facility” on page 357, from a backed up version of the file.

You can use third-party software to backup Model 204 files. Third-party backups run independently of Model 204, while the Online remains running and responsive to non-updating activity. This provides a greater ability to run Model 204 in a nonstop 24*7 mode.

The following describes the functionality and facilities that allow you to develop, integrate, and manage your third-party software backup methodology.

Managing third-party backups

As third-party software is unaware of Model 204 file management, it must be assured that when a file is being backed up that the file is logically and physically consistent. An external backup of a file taken when Online updates are occurring is unreliable and must not be used in any subsequent attempt to restore or recover the file. To assure file integrity, all updating must cease and remain stopped until the backup processing is complete.

Understanding an extended quiesce

You can create a period of time during which files cannot be updated. By using a set of commands, two CCAIN parameters, and a set of \$functions, you can create an interval, after a checkpoint is taken, during which updating is stopped. This interval is called the extended quiesce interval or, simply, an **extended quiesce**.

User Language file updating requests go into a bumpable wait when the system is in the extended quiesce state. If the users in such a wait are bumped, their threads are restarted with the following message:

```
M204.2546:SOFT RESTART OF USER DUE TO BUMP WHILE IN  
EXTENDED QUIESCE
```

File commands that start update units remain in a nonbumpable, swappable wait until the extended quiesce ends. File updating IFAM threads remain in nonbumpable, swappable waits during extended quiesce.

Signing on IFAM threads during an extended quiesce

When single or multicursor IFAM threads signs on to Model 204, they issue the following message: M204.0962 SIGN ON, JOB NAME = XXXXXXXX'. During an extended quiesce, IFAM2 threads (IODEV=23) cannot sign on to Model 204. If these threads try to sign on during extended quiesce, they are placed in a swappable wait of type 20. The sign-on messages are not displayed. The wait ends when the extended quiesce terminates. Then, the

waiting threads can sign on. When signing on is complete, the MONITOR and IFAMSTAT commands can display their status

Coordinating an external backup

External backups must be taken only during an extended quiesce. The duration of the extended quiesce depends on when a CHECKPOINT END EXTENDED QUIESCE command is issued or, on the values of the parameters, CPQZSECS and CPQZACTN. For more discussion of the parameters, see CPQZACTN: Action to take when the CPQZSECS time limit expires and CPQZSECS: Maximum duration, in seconds, of an extended quiesce in the *Model 204 Parameter and Parameter and Command Reference*.

You can create and manage an extended quiesce with the following System Manager commands:

- `CHECKPOINT SET Extended Quiesce`
When issued, this command places the Online into an extended quiesce immediately after the next successful checkpoint.
- `CHECKPOINT UNSET Extended Quiesce`
The UNSET option reverses the effect of the SET option.
- `CHECKPOINT END Extended Quiesce`
This command terminates an extended quiesce.

For more discussion of these CHECKPOINT commands, see the *Rocket Model 204 Parameter and Parameter and Command Reference*.

Ring stream journals and extended quiesce

If a ring stream journal is in use, an off-loading is automatically attempted at the beginning of the extended quiesce. If off-loading is successful, the next ring stream member is made active. At the end of the extended quiesce, a checkpoint is taken and a checkpoint record is written on the active journal member. If a subsequent RESTART recovery is required, the active journal member will contain a checkpoint record.

Programming a third-party backup

The automated submission of third-party backup jobs requires a User Language interface to the internal extended quiesce data structures. This is provided primarily by using the ECB-related \$functions, \$ECBDGET, \$ECBDSET, \$POST, \$UNPOST, and \$WAIT as well as the named ECBs used for extended quiesce: CPQZ and QZSIG. Both ECBs are defined internally to the checkpoint subsystem.

- The CPQZ ECB is automatically posted at the start of an extended quiesce and automatically unposted at the end of an extended quiesce.

By specifying CPQZ as a keyword, you can use the \$ECBDGET and \$ECBDSET functions to write and read up to 255 bytes of data to and from a checkpoint-maintained internal area. This data area is cleared automatically at the end of extended quiesce.

- A User Language thread can post the QZSIG ECB by issuing a \$POST function only during an extended quiesce.

Another User Language thread can wait on the posting of the QZSIG ECB, but only during an extended quiesce interval. Outside an extended quiesce, the QZSIG ECB is unposted.

See the chapter on User Language functions in the *Rocket Model 204 User Language Manual* to examine the syntax and use of the \$ECBDGET, \$ECBDSET, \$ECBTEST, \$POST, \$UNPOST and \$WAIT functions.

Programming a wait for extended quiesce

The following User Language procedure demonstrates the use of the extended quiesce and the named ECBs to submit and control a third-party back-up job.

```
PROCEDURE WAIT.FOR.EXTENDED QUIESCE
BEGIN
IF $SETG('HDRCTL',$VIEW('HDRCTL')) THEN
PRINT 'GTBL NOT SET - BACKUP WILL NOT BE SUBMITTED'
STOP
ELSE
PRINT '$WAIT CPQZ ' with $WAIT('CPQZ')
END IF
END
R HDRCTL 3
USE $JOB
DISPLAY BACKUP.MODEL204.DATABASE.FILES.JOB
R HDRCTL ?&HDRCTL
BEGIN
PRINT '$WAIT QZSIG - WAIT UNTIL BACKUP COMPLETES ' -
      WITH $WAIT('QZSIG')
IF $ECBDGET('CPQZ') = 'GOOD BACKUP' THEN
PRINT 'BACKUP SUCCESSFUL'
ELSE
PRINT 'BACKUP UNSUCCESSFUL'
END IF
END
CHECKPOINT END EXTENDED QUIESCE
END PROCEDURE
```

Once the extended quiesce starts, the previous User Language request continues to the \$WAIT('QZSIG') function call and waits for third-party back-up job to end.

Submitting a backup

In the following procedure, BACKUP.MODEL204.DATABASE.FILES.JOB, the BACKUP step executes an IEFBR14—essentially a no-operation. However, this job could be the submission of any third-party backup software that meets your backup requirements. The subsequent BATCH2 steps are executed depending upon the condition code from the step named BACKUP.

```

PROCEDURE BACKUP.MODEL204.DATABASE.FILES.JOB
//PLACE A SITE-VALID JOBCARD HERE
//BACKUP EXEC PGM=IEFBR14,COND=EVEN
//DD1 DD DISP=SHR,DSN=YOURDATASET.TOBACKUP
//* ITFAILED STEP IF BACKUP DID NOT END WITH CONDITION CODE 0
//* EOJ THE ONLINE
//ITFAILED EXEC PGM=BATCH2,PARM='CRIOCHNL',COND=(0,EQ,BACKUP)
//STEPLIB DD DSN=MODEL204.LOADLIB,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CCAOUT DD SYSOUT=*
//CCAIN DD *
LOGIN SUPERKLUGE
PIGFLOUR
BEGIN
    PRINT 'PASS INFO TO $WAIT QZSIG WAITER ' -
        WITH $ECBDSET('CPQZ','BAD BACKUP')
    PRINT '$POST QZSIG - WAKE UP $WAIT QZSIG WAITER ' -
        WITH $POST('QZSIG')
    END
EOD
EOJ
LOGOUT
//* ITWORKED STEP IF BACKUP DID COMPLETE WITH CONDITION CODE 0
//* END THE EXTENDED QUIESCE IN THE ONLINE
//ITWORKED EXEC PGM=BATCH2,PARM='CRIOCHNL',COND=(0,NE,BACKUP)
//STEPLIB DD DSN=MODEL204.LOADLIB,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CCAOUT DD SYSOUT=*
//CCAIN DD *
LOGIN SUPERKLUGE
PIGFLOUR
BEGIN
    PRINT 'PASS INFO TO $WAIT QZSIG WAITER ' -
        WITH $ECBDSET('CPQZ','GOOD BACKUP')
    PRINT '$POST QZSIG - WAKE UP $WAIT QZSIG WAITER ' -
        WITH $POST('QZSIG')
    END
LOGOUT
/*
//

```

END PROCEDURE

Managing the extended quiesce

In the previous example, should a CPQZSECS time-out cause a CPQZACTN argument to issue an internal CHECKPOINT END EXTENDED QUIESCE command, the ending of the extended quiesce is delayed until the thread, which issued the \$WAIT('QZSIG'), drops out of wait (wait type 48). The \$WAIT('QZSIG') can be satisfied by one of the following:

- Have another thread issue a \$POST('QZSIG') function call
- Bump the thread in wait 48
- Issue an EOJ command

Use of the \$WAIT('QZSIG') function is an aid to prevent the inadvertent ending of an extended quiesce while a third-party backup job is running.

Note: A \$POST('QZSIG') function call can be issued only once per extended quiesce. A \$ECBDSET function call may not be issued after a \$POST('QZSIG') function has been issued.

The previous code examples perform no backups and are not intended as full representations of what you can or should use as your backup strategy.

BATCH2 backup requirements

To use a BATCH2 connection to notify or signal the Online as to how to proceed when your backup processing is complete, the Online needs:

- BATCH2 IODEV definition
- CRAM installed and activated
- CRIOCHNL defined in the CCAIN stream, if it is not using the default CRIOCHNL name.

Delayed ending for extended quiesce or status message

During an extended quiesce, the following message is displayed on the operator's console every minute, or the CPQZSECS time limit if it is less than 60.

```
M204.2610. SYSTEM ENTERED EXTENDED QUIESCE AT: mm/dd/yy
hh:mm:ss, ALL FILE UPDATING REMAINS SUSPENDED { -- END
EXTENDED QUIESCE PROCESSING IS DELAYED WITH nnnn USER(S)
IN WAIT TYPE 47, mmmmm USER(S) IN WAIT TYPE 48 }
```

When the extended quiesce ends, one of the versions of M204.2613 message is displayed on the operators console.

```
M204.2613. SYSTEM ENTERED EXTENDED QUIESCE AT: mm/dd/yy
hh:mm:ss, SYSTEM EXITED EXTENDED QUIESCE AT: mm/dd/yy
hh:mm:ss, REASON = reason-text
```

where *reason-text* is one of the following:

- END COMMAND ISSUED BY USER: nnnnnn
- EOJ COMMAND
- CPQZACTN TIME OUT ACTION
- CHKPPST RESTART
- Issuing a CHKMSG or MONITOR CHECKPOINT command during extended quiesce displays the M204.2610 message, in addition to other CHKMSG or MONITOR CHECKPOINT command information.
- Issuing a CHKMSG or MONITOR CHECKPOINT command after an extended quiesce ends displays a M204.2613 message with other checkpoint information.

If an extended quiesce is to end, because either a CHECKPOINT END EXTENDED QUIESCE command was issued, or X'40' is specified for ENDEQ as the CPQZACTN argument at CPQZSECS time-out, and there is a User Language thread in \$WAIT('QZSIG') wait, the M204.2610 message is repeated on the operators console every minute as follows:

```
M204.2610. SYSTEM ENTERED EXTENDED QUIESCE AT: yy/ddd
hh:mm:ss, ALL FILE UPDATING REMAINS SUSPENDED -- END
EXTENDED QUIESCE PROCESSING IS DELAYED WITH nnnn USER(S)
IN WAIT TYPE 48.
```

This form of the M204.2610 message can also be displayed with the CHKMSG or MONITOR CHECKPOINT commands.

Using the UTILJ utility

Use the standalone, batch utility, UTILJ, for the following purposes:

- Create end-of-file (EOF) markers on journal data sets that were never closed because of a system failure
- Obtain information about the contents of journal data sets for performance analysis, capacity planning, and determining the optimal number and size of journal buffers
- As a debugging utility when working with Rocket Software Technical Support

UTILJ and stream configurations

Because the UTILJ utility processes only single data sets, the following considerations apply:

- When creating EOF markers, copy members of concatenated, parallel, or ring streams using the COPY STREAM command described in the *Model 204 Parameter and Command Reference*. In this manual, recovery data streams are discussed.
- When generating reports, process members of concatenated, parallel, or ring streams one at a time. To report on the contents of a concatenated stream, use regular z/OS concatenation.

UTILJ options

The following options control the execution and output of the UTILJ utility:

Table 16-3. UTILJ utility options

Option	Specifies	Description
DISPDUPS	To override the suppression of duplicate output lines when displaying hexadecimal output.	
EOF	Input is a journal without an end-of-file marker	You cannot specify other options with the EOF option. You must have write access to the device containing the journal to be repaired, except in z/VSE environments. See also “EOF option” on page 395.
FDIR	File directory number of the run creating the journal	Prints all information related to the specified file unless further qualified by other parameters such as FROMTIME, TOTIME, and RECTYPE.
FILENAME	DDNAME of the file, the name specified in the CREATE command, to be printed	All file-related entries are printed unless further qualified by other parameters, such as FROMTIME, TOTIME, and RECTYPE.
FORMAT	Both report types 1 and 2 are to be printed	This option is retained for compatibility with previous releases.
FROMDATE and TODATE	Range of dates for which journal records are printed	Must be 5-digit Julian dates such as 02062. For behavior, see Table 16-4 on page 396
FROMTIME and TOTIME	Time range for which journal records are printed	Must be in 24-hour clock (hhmmss) format such as 151003 = 3:10:03 P.M. For behavior, see Table 16-4 on page 396

Table 16-3. UTILJ utility options (Continued)

Option	Specifies	Description
RECTYPE	Type of records	Repeat RECTYPE as necessary. The default is all record types. For information about journal record types, see Appendix A.
REPORT	Type of report to print	Discussed separately in “UTILJ REPORT options” on page 396
START	Update unit number with which to begin printing.	
STOP	Update unit number at which to end printing.	
SUBTYPE	Subtype of the records printed	Typically, this option is used with RECTYPE = 6 to restrict the type of ROLL FORWARD entries printed. SUBTYPE values can be found in Appendix A, where journal layouts are documented. UTILJ expects a SUBTYPE value to be entered in decimal format, and so, for example, a SUBTYPE of x'81' (since-last statistics including conflict statistics) should be entered using its decimal value, SUBTYPE=129.

EOF option

The EOF option writes an end-of-file marker at the appropriate place. The UTILJ utility reads each block of the CCAJRNL data set until it encounters an I/O error or a validation error, such as a time stamp out of sequence. The journal is then physically closed, causing an end-of-file marker to be written.

A journal that is part of a normal Model 204 run that has an end-of-file marker present is not affected when used as input to the UTILJ utility.

From-to date-time options

The following table describes the behavior of UTILJ, depending on the combination of the from-to-date options and from-to-time options.

Table 16-4. From-to date-time option combinations

Option present	And option absent	Read input journal...
FROMDATE / TODATE	FROMTIME / TOTIME	From beginning to end; process any record with a date in the range of FROMDATE to TODATE inclusive ignoring the time.
FROMTIME / TOTIME	FROMDATE/TODATE	From beginning to end; process any record with a time in the range of FROMTIME to TOTIME inclusive ignoring the date.
FROMTIME	TOTIME	As if TOTIME was 2359
TOTIME	FROMTIME	As if FROMTIME was 0000
FROMDATE	TODATE	From beginning to end; process any record with a date greater than or equal to FROMDATE value until the end of the input journal.
TODATE	FROMDATE	From beginning to end; process all records from beginning of the journal with a date less than or equal to TODATE value.

UTILJ REPORT options

UTILJ reports have a 2-line header, as shown in the following example, that reflects the circumstances of your site.

```
UTILJ UTILITY - VERSION 7.1.0D
DATE/TIME OF RUN: 03/31/2009 00:18:48
```

The REPORT options for the UTILJ utility are:

Option	Prints...
1	Only audit trail information, no hexadecimal
2	Individual journal records formatted in only hexadecimal
4	Journal blocks in only hexadecimal
8	Histogram showing the size ranges of the blocks written in a run, which is helpful in determining size and number of buffers.

Using CCAJLOG and option 1

If CCAJLOG is in use, Report option 1 selects no output since audit information is absent from the CCAJRNL.

Using Options 2, 4, and 8

A histogram is particularly useful when there are frequent small updates or in a read-only environment where the buffer is written infrequently.

Report option values are hexadecimal and can be summed. For example, to combine options 2 and 8, specify REPORT=A (not 10, which generates an invalid function error).

The default is REPORT=F, specifying to print all report types.

For example:

BLOCKSIZE RANGE IN BYTES	COUNT
6144 - 6655	224
5632 - 6143	1020
5120 - 5631	196
4608 - 5119	212
4096 - 4607	224
3584 - 4095	252
3072 - 3583	264
2560 - 3071	372
2048 - 2559	676
1536 - 2047	1300
1024 - 1535	2600
512 - 1023	5128
0 - 511	25244
TOTAL # BLOCKS	37712
DCB BLOCKSIZE IS	6749

- *BLOCKSIZE RANGE IN BYTES* is the breakdown of all block sizes, up to the largest block written. Unused space in the buffer is not listed.
- *COUNT* is the number of blocks in that *BLOCKSIZE* range read during the scan of CCAJRNL.
- *TOTAL # BLOCKS* is the number of blocks read during the scan of CCAJRNL.
- *DCB BLOCKSIZE* is the size of each journal buffer. In this example, the buffer is the default size, 6749 bytes.

UTILJ return codes

The return codes in Table 16-5 apply to the UTILJ utility.

Table 16-5. UTILJ return codes

Return code	Description
0	Normal return
16	OPEN failed for CCAPRINT
20	OPEN failed for CCAJRNL
24	I/O error occurred reading or writing CCAJRNL
28	CCAJRNL contains one or more corrupted journal blocks

Return code 28 is issued whenever a corrupted journal block is detected while reading CCAJRNL. The following message is also printed at the end of LISTING:

```
***** THIS CCAJRNL CONTAINS ONE OR MORE CORRUPTED JOURNAL  
BLOCKS - PLEASE CONTACT CCA CUSTOMER SUPPORT
```

This is a serious error that must be brought to the immediate attention of Technical Support.

z/VSE considerations

In z/VSE, print or copy options are specified through a control statement from SYSIPT:

- Omission of the control statement produces a hexadecimal dump.
- The control statement format is free form, with continuation from one input statement to the next specified by a comma immediately following the last parameter on the statement to be continued.
- The input stream defined in the JCL can be either a DLBL and EXTENT or a TLBL for file name CCAJRNL.

If the input file is on tape, symbolic unit SYS004 must be assigned to the tape drive on which the tape is mounted.

- The EOF option also requires an output stream, which must be defined in the JCL by either a DLBL and EXTENT or a TLBL for file name CCARF.

If the output file is on tape, symbolic unit SYS005 must be assigned to the tape drive on which the tape is mounted.

UTILJ examples

The following examples provide the JCL to run UTILJ and illustrate how to request the printing of journal information for the hour between 12:05:13 and 13:05:13.

z/OS UTILJ example

```
//UTILJ EXEC PGM=UTILJ,PARM='FROMTIME=120513,TOTIME=130513'
//STEPLIB DD DSN=M204.LOADLIB,DISP=SHR
//CCAPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//CCAJRNL DD DSN=M204.JOURNAL,DISP=OLD
```

z/VSE UTILJ example

In this example, UTILJ transfers journal streams between disk and tape media, or prints the journal in hexadecimal format. You can select only one option for each execution of UTILJ:

```
// JOB UTILJ PRINT A JOURNAL IN HEX DUMP AND FORMATTED MODE
// DLBL CCAJRNL,'CCAJRNL.FILE',0
// EXTENT SYS001,SYSWK1,,,1000,390
// EXEC UTILJ,SIZE=AUTO
REPORT=2,
FROMTIME=120513,TOTIME=130513
/*
/&

// JOB UTILJ COPY A JOURNAL FROM DISK TO TAPE
// DLBL CCAJRNL,'CCAJRNL.FILE',0
// EXTENT SYS001,SYSWK1,,,1000,390
// TLBL CCARF,'CCAJRNL.BACKUP'
// ASSGN SYS005,X'300'
// EXEC UTILJ,SIZE=AUTO
REPORT=2,
FROMTIME=120513,TOTIME=130513
EOF
/*
/&
```

CMS UTILJ example

Use the UTILJ EXEC procedure to run the UTILJ utility under CMS:

```
UTILJ destination dsn mode (FROMTIME 120513 TOTIME 130513
```

Where

- *destination* specifies the location of the print report where T=*user terminal*, and D = *user A-DISK*. A-DISK is the default. The report appears on the disk as:

```
UTILJ NEWLST A
```

- *dsn* is the data set name. For z/OS format data sets, the DSN is specified with spaces instead of periods between the qualifiers. A z/OS format journal data set is specified in the FILEDEF with spaces between the qualifiers. For CMS files, the DSN is specified as file name and filetype.
- *mode* specifies the access mode of the disk containing the journal file to be processed.

For tape files, the DSN and MODE parameters are replaced by the keyword TAPE. Tape files also require issuing a FILEDEF (and a LABELDEF, if necessary) for the checkpoint data set before issuing the UTILJ command. For example:

```
FILEDEF CCAJRNLTAP1 SL VOLID 12345 (RECFM U LRECL 0 BLKSIZE 6176
LABELDEF CCAJRNLT standard labeldef parameters
UTILJ D TAPE (FROMTIME 120513 TOTIME 130513
```

Using UTILJ to analyze problems

UTILJ lets you dump records from CCAJRNLT for ease in debugging problems. If you call Technical Support for assistance in debugging problems, you may be asked to run JCL similar to the following.

```
//UTILJ EXEC PGM=UTILJ, PARM='FROM-
TIME=123500, TOTIME=124500'
//STEPLIB DD DISP=SHR, DSN=xxxxxxxxxxx.LOADLIB
//SYSUDUMP DD SYSOUT=*
//CCAPRINT DD SYSOUT=*
//CAJRNLT DD DISP=SHR, DSN=xxxxxxx.CAJRNLT
//CAOUT DD DSN=xxxxxxxxxxx.CAOUT,
// DISP=(NEW,CATLG,CATLG), UNIT=3390,
// SPACE=(TRK,(500,100),RLSE)
//
```

Defining CAOUT for additional diagnostics

You can allocate a CAOUT DD in z/VM or z/OS. The CAOUT data set that is created makes it easy for developers and/or Technical Support to inspect journal data.

When CAOUT is specified, only the date and time option pairs—FROMDATE/TODATE and FROMTIME/TOTIME are valid for controlling data into CAOUT. If the FROMTIME/TOTIME parameters are used to limit the number of journal records extracted, the CAOUT data set will typically be much smaller than the CAJRNLT data set.

If CCAOUT is specified, journal records are not written to CCAPRINT. Journal records are unblocked and written individually to CCAOUT in hexadecimal format. Allocate a CCAOUT DD only when you are selecting a subset of a journal for subsequent UTILJ or recover processing by Rocket Software.

To extract a single file's worth of entries from a larger journal, specify the FILENAME=*filename* option for UTILJ. This option selects all journal records that are associated with *filename* for copying into the CCAOUT DD.

Using the MERGEJ utility

Merging overlapping journal files

The MERGEJ utility, available in z/OS, z/VSE, and z/VM environments, is used to merge journals produced by different Model 204 runs into a single file. Journals created by jobs that ran concurrently and journals that do not contain EOF markers must be merged before being used as input for a media recovery run. (For information on the creation of EOF markers, see the section "Using the UTILJ utility" on page 393).

Journal files run against MERGEJ can be individual files or the output of a previous MERGEJ run.

Note: Journals that do not overlap do not need to be merged. In this case, the individual journals can be concatenated for input to recovery and MERGEJ would not be required.

MERGEJ support for journal types

The MERGEJ utility can determine the journal type being merged. If the data set type is unknown, MERGEJ sets a return code of 2 and issues the following warning message.

```
*** WARNING: UNKNOWN DATASET TYPE: PLEASE SPECIFY TYPE VIA
PARAMETER SETTING OF JRNL, JLOG OR BOTH.
```

A merged journal may contain audit information (CCAJLOG) and/or recovery (CCAJRNL) information. A data set containing only:

- Recovery information cannot be used with AUDIT204.
- A data set containing only messages and audit information cannot be used for REGENERATE or RESTART.

If the MERGEJ utility attempts to use CCAJRNL or CCAJLOG data sets that are not the first data set for each DD respectively (such as, GDG members for CCAJRNL that are the fourth, sixth, and so on, data sets), the initial record is not present and the type of data, JRNL or JLOG, cannot be determined. The MERGEJ utility terminates with a return code of 2 indicating unknown data set type.

To inform the MERGEJ utility as to the type of data to be processed set the PARM parameter to JRNL, JLOG, or BOTH. For example, in z/OS specify:

```
EXEC PGM=MERGEJ, PARM=BOTH
```

Examining PARM arguments

Table 16-6 on page 402 shows what happens with various combinations of the PARM value on the MERGEJ job and the types of input (JLOG and/or JRNL). Following Table 16-6 the messages listed in the Message column are fully expanded. The value of the PARM statement (using BOTH, JRNL, or JLOG) is how the merged journal is marked.

- A merged journal marked as BOTH can be read by both Audit204 and REGEN.
- A merged journal marked as JRNL and used as input to Audit204 always gets a return code of 20 and message:

```
M204.2515: NON-MESSAGE DATASET IS INVALID FOR CCAJRNL.
```

- A merged journal marked as JLOG and used as input to regenerate always gets a return code of 4 and message:

```
M204.2515: MESSAGES ONLY DATASET IS INVALID FOR CCAGEN
```

Table 16-6. return codes from Merge/Audit/Regen

	PARM	INPUT	M	A	R	Step	Message
A	BOTH	JLOG1/JRNL1	0	0	0	R)	Regenerated file updates from first run
A1	BOTH	JLOG2/JRNL	0	0	0	R)	Regenerated file updates from first run
B	BOTH	JLOG1/JLOG2	2	0	4	M)	** ...PARM=BOTH ... ONLY JLOG...
						R)	2128: THERE WERE NO UPDATES TO FILE ...
C	BOTH	JRNL1/JRNL2	2	0	0	M)	** ...PARM=BOTH ... ONLY JRNL...
D	JRNL	JRNL1/JRNL2	0	20	0	A)	2515: NON-MESSAGE...
						R)	Regenerated file updates from both runs
E	JRNL	JLOG1/JLOG2	2	20	4	M)	** ...PARM=JRNL ... ONLY JLOG...
						A)	2515: NON-MESSAGE...

Table 16-6. return codes from Merge/Audit/Regen

	PARM	INPUT	M	A	R	Step	Message
						R)	2128: THERE WERE NO UPDATES TO FILE ...
F	JRNL	JLOG1/JRNL2	2	20	4	M)	** ...PARM=JRNL ... BUT ALSO JLOG...
						A)	2515: NON-MESSAGE...
						R)	1406: MISSING JOURNAL WAS DETECTED ...
G	JLOG	JLOG1/JRNL2	2	0	4	M)	* ...PARM=JLOG ... BUT ALSO JRNL...
						R)	2515: MESSAGES ONLY...
G1	JLOG	JRNL1/JLOG2	2	0	4	M)	** ...PARM=JLOG ... BUT ALSO JRNL...
						R)	2515: MESSAGES ONLY...
H	JLOG	JLOG1/JLOG2	0	0	4	R)	2515: MESSAGES ONLY...
I	JLOG	JRNL1/JRNL2	2	0	4	M)	** ...PARM=JLOG ... BUT ONLY JRNL...
						R)	2515: MESSAGES ONLY...
J	None	JRNL1/JLOG1	0	0	0	R)	Regenerated file using JRLN1
K	None	JLOG1/JLOG2	0	0	4	R)	2515: MESSAGES ONLY...
L	None	JRNL1/JRNL2	0	20	0	A)	2515: NON-MESSAGE...
						R)	Regenerated file using JRNL1

The complete messages listed in the Table 16-6 Message column:

** WARNING: PARM=BOTH SPECIFIED BUT ONLY JLOG INPUT WAS
FOUND.

** WARNING: PARM=JLOG SPECIFIED BUT ALSO JRNL INPUT WAS
FOUND.

** WARNING: PARM=JRNL SPECIFIED BUT ONLY JLOG INPUT WAS
FOUND.

M204.1406: MISSING JOURNAL WAS DETECTED BETWEEN dd mmm yyyy
hh:mmth AND dd mmm yyyy hh:mmth FOR FILE filename

M204.2128: THERE WERE NO UPDATES TO FILE filename IN CCAGEN

M204.2515: NON-MESSAGE DATASET IS INVALID FOR CCAJRNL

M204.2515: MESSAGES ONLY DATASET IS INVALID FOR CCAGEN

MERGEJ considerations

Keep the following considerations in mind when using MERGEJ:

- If a previously merged file is combined with individual journal files, the merged file must be the first input file specified (SORTIN01).
- When the MERGEJ utility is run, the following error conditions are detected:

Error condition	MERGEJ behavior
Missing input journal file(s)	Terminates without performing a merge.
Missing output journal file	Terminates without performing a merge.
I/O error on input or output data sets	Terminates without performing a merge.
Merged journal used as input for other than the SORTIN01 data set (SORTIN1 for DOS)	Terminates without performing a merge.
Sequence error: a journal file not closed properly and the EOF marker is missing	Assumes the presence of an EOF marker and proceeds with processing.

You cannot use streams as input to MERGEJ, with the exception of GDG streams—see “Using MERGEJ with GDG streams” on page 407.

Use the COPY command to copy journal streams into individual data sets before using them as input to the MERGEJ utility.

Data sets required for the MERGEJ utility

To run the MERGEJ utility, you must define the following data sets:

- z/OS and CMS operating systems
 - SORTIN01 through SORTIN nn , sequentially numbered, not to exceed SORTIN32, which define the journal files from the original updating Model 204 runs being recovered as input to the MERGEJ utility
 - SORTOUT to define the single output file
 - Standard IBM or IBM-compatible SORTLIB and SORTMSGs data sets
- For the z/VSE operating system
 - SORTIN1 through SORTIN n , sequentially numbered, not to exceed SORTIN9, which define the journal files from the original updating Model 204 runs being recovered as input to MERGEJ
 - SORTOUT to define the single output file

- DLBL and LIBDEF statements pointing to the library and sublibrary containing the IBM or IBM-compatible SORT or MERGE program

Each SORTIN deata set should represent the entire journal for a single job. For example, if Job1 creates Journal1, Journal2, and Journal3 and Job 2 creates Journal4, the following illustrates how the SORTIN data sets should be set up:

```
//SORTIN01 DD DSN=LOCAL.M204.JOURNAL1,DISP=SHR
//          DD DSN=LOCAL.M204.JOURNAL2,DISP=SHR
//          DD DSN=LOCAL.M204.JOURNAL3,DISP=SHR
//SORTIN02 DD DSN=LOCAL.M204.JOURNAL4,DISP=SHR
```

The following examples illustrate further operating system-specific considerations.

z/OS MERGEJ example

```
// EXEC PGM=MERGEJ
//STEPLIB DD DSN=LOCAL.M204.LOADLIB,DISP=SHR
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//SORTOUT DD DSN=LOCAL.M204.MERGED.JOURNAL.J840302,DISP=OLD
//SORTIN01 DD DSN=LOCAL.M204.JOURNAL.840301,DISP=SHR
//SORTIN02 DD DSN=LOCAL.M204.JOURNAL.840302,DISP=SHR
```

z/VSE MERGEJ example

```
// JOB MERGEJ
// DLBL M204LIB,'M204.PROD.LIBRARY'
// EXTENT ,volser
// LIBDEF PHASE,SEARCH=M204LIB.R210
// TLBL SORTOUT
// ASSGN SYS020,cuu
// DLBL SORTIN1,'M204.CCAJRN1.1',,SD
// EXTENT SYSnnn,balance of extent information
// DLBL SORTIN2,'M204.CCAJRN1.2',,SD
// EXTENT SYSnnn,balance of extent information
// TLBL SORTIN3
// ASSGN SYS023,cuu
// DLBL SORTIN4,'M204.CCAJRN1.4',,SD
// EXTENT SYSnnn,balance of extent information
// TLBL SORTIN5
// ASSGN SYS025,cuu
// TLBL SORTIN6
// ASSGN SYS026,cuu
// EXEC MERGEJ,SIZE=(AUTO,xxK)
/*
/&
```

Using MERGEJ in z/VSE

In z/VSE, use the MERGEJ utility:

- If using tape for the SORTOUT or any of the SORTINx data sets, the SYS number is fixed.
- You must set the SYS number of the SORTOUT data set to 020, for example:

```
// ASSGN SYS020, cuu
```

- You must set the SYS number of the SORTIN data set to 020 plus the number of the SORTIN data set. For example, SORTIN1 = SYS021, SORTIN5 = SYS025, and SORTIN9 = SYS029.
- The SIZE parameter on the // EXEC card is mandatory.
 - AUTO specification reserves storage for the MERGEJ program.
 - xxK reserves storage for the SORT program, which is dynamically loaded during execution. To determine the amount of additional storage needed for the sort program, refer to the IBM *SORT/MERGE Programmer's Guide* or equivalent manual if using a sort package other than IBM.
- You can use conditional job control. The return code is set on the program termination.

Using MERGEJ in CMS

In CMS, the MERGEJ EXEC runs the MERGEJ utility. The command format is:

```
MERGEJ fileid / fileid / ... / outfileid
```

Where

- *fileid* specifies the file identifier and mode of a journal file.

A file identifier can be either the name of a data set on a variable-format disk or the name and type of a file on a CMS-format disk. If you specify a data set name, insert a white space to separate the qualifiers.
- *outfileid* is a CMS file that specifies the identifier and mode of the file that is to contain the merged journal file.

The MERGEJ command requires the availability of a dynamically invoked SORT command. The number of journal files that can be merged with one MERGEJ command is a function of the capacity of the SORT/MERGE utility used. For example:

```
MERGEJ JOURNAL F /JOURNAL F/ MERGED JOURNAL A
```


Using MERGEJ with GDG streams

If the journal is defined as a GDG stream in the Online run, each generation created during the run is considered only a part of the journal. The complete journal consists of all the generations that were produced during that run.

When you run MERGEJ, each SORTINnn DD should consist of a complete journal. For example, if the complete journal from the first run consists of generations 54 and 55 and the complete journal from the second run consists of generations 56 through 58, the following would be used as input to MERGEJ:

```
//* complete journal from Tuesday:
//SORTIN01 DD DISP=SHR,DSN=xxx.CCAJRNL.G0054V00
//          DD DISP=SHR,DSN=xxx.CCAJRNL.G0055V00
//* complete journal from Wednesday:
//SORTIN02 DD DISP=SHR,DSN=xxx.CCAJRNL.G0056V00
//          DD DISP=SHR,DSN=xxx.CCAJRNL.G0057V00
//          DD DISP=SHR,DSN=xxx.CCAJRNL.G0058V00
```


17

Configuring Checkpoint and Journal Data Streams

In this chapter

- Overview
- DEFINE STREAM command
- Ring stream configuration
- Ring streams and recovery
- Concatenated stream configuration
- Parallel stream configuration
- Perpetual journaling for z/OS
- Example 1: Ring/parallel journal stream
- Example 2: Parallel checkpoint stream
- Example 3: Concatenated tape off load stream for z/OS
- Troubleshooting GDG streams using specific generations

Overview

The Model 204 sequential file processing facility defines and processes checkpoint and journal data streams. Using the logical I/O stream configurations available through the DEFINE STREAM command,

recovery procedures can assure a complete and consistent database across a system failure.

The concatenated and ring configurations help overcome physical hardware and operating system limitations. The parallel configuration allows multiple data set copies. You can use concatenated, parallel, and ring configurations separately or in any combination for checkpoint and journal streams.

Sites with z/OS can also define the checkpoint and journal data streams with GDG data sets. GDG data sets can be the only configuration for your checkpoint and journal data streams or they can be part of your ring, concatenated, or parallel configurations.

This chapter describes each sequential I/O processing configuration.

DEFINE STREAM command

The Model 204 sequential file processing facility—DEFINE STREAM command—ensures a complete and consistent database across a system failure. The DEFINE STREAM command:

- Logs journal and checkpoint files to sequential data sets
- Automatically switches to alternate data sets on a file-full condition (concatenated and ring stream configurations)
- Writes multiple copies simultaneously (parallel stream configuration)
- Automatically creates a tape backup of checkpoint and journal files (ring stream configuration with tape offload)
- Automatically opens a sequentially numbered data set as needed—GDG data set.

You can copy the contents of a stream to another stream or single data set using the COPY command.

For examples and further explanation of the DEFINE STREAM command, see “Example 1: Ring/parallel journal stream” on page 421.

Using the DEFINE STREAM command

The following considerations apply to DEFINE STREAM commands:

- You must specify DEFINE commands before User 0 input, because both the journal and checkpoint streams are opened before the User 0 input line is processed.
- The order of DEFINE commands is not important. The definition of the stream is resolved at open time.
- Members of a stream are reserved when the stream is open, preventing other Model 204 users from opening them.

- If a ring stream is defined, to prevent exhausting the journal stream capacity, the members in the ring are offloaded to tape as they fill up.
- The control stream is a very small data set and no DCB attributes are required for its allocation.

Note: Do not use the prefix TAPE; it is a reserved prefix used by deferred update data sets. Using this prefix with a dynamically allocated offload stream can cause problems with checkpoint ring streams.

To set up a DEFINE STREAM command with the options suitable to your site, see the *Rocket Model 204 Parameter and Command Reference*.

Off load and control streams overview

The **off load stream** can be any stream configuration. The next section provides details on offload processing.

Offload is performed via a Model 204 pseudo subtask (PST), if there is a PST available when the offload process begins. The number of subtasks specified on User 0's parameter line (NSUBTKS) must specify enough subtasks to include all offload processes that begin during Model 204 operation, such as one subtask for each ring stream defined in the run. If the PST cannot be started, the offload process is done synchronously, using the current user's server.

Pseudo subtasks are discussed in "Pseudo subtasks" on page 132. NSUBTKS is described in Table 2-4 on page 27.

The **control stream** is used during the open process when the ring or GDG stream is used for input. Data on the control stream provides the ability to recreate the status of the ring or GDG stream at the time of its last use as an output stream.

Ring stream configuration

Ring streams reduce the possibility of filling up journal or checkpoint data sets by processing them cyclically. When a ring stream member is full, it is offloaded to separately defined offload data sets. After the offload is complete, the ring member can be written to again as the checkpoint or journal data set. In addition to ring members and offload data sets, a control stream also maintains ring status information.

Ring stream configurations make it possible to write large amounts of journal and checkpoint data without bringing down a Model 204 Online due to space constraints. For this reason, ring journaling is sometimes referred to as perpetual journaling.

Note: Long Online runs involve a trade-off between continuous access and recovery time. When recovery processing searches for checkpoints in offload data sets, recovering a large ring stream configuration can take a long time.

Therefore, Rocket Software recommends bringing down an Online periodically even if ring stream configurations are used.

The use of ring streams for checkpointing is compatible with the CPMAX parameter described in Table 15-1 on page 333.

For DEFINE STREAM and job control examples, see the section “Example 1: Ring/parallel journal stream” on page 421.

Output ring stream processing sequence

Output ring stream processing for checkpoints and journals follows these steps:

1. Records are written to one of the members of the cyclic group.
2. When a member stream is filled, it is closed and scheduled for asynchronous offloading.

Offloading starts automatically when a user-specified (AUTOOFFLOAD) number of members become full and continues until all full members of the ring are offloaded. The auto-offload threshold cannot exceed the number of ring members minus one.

3. The next member of the ring is opened and output processing to the ring continues.
4. The stream is considered full when all members of the ring become full and the offload stream is full, or when the control stream becomes full.
5. The offload stream is switched when no members are left to offload.

Note: Dynamic allocation of ring stream members can cause unpredictable results during member switching.

Types of off load streams

The offload stream in a ring configuration can be any type: concatenated, ring, or parallel. Concatenated and parallel streams are explained later in this chapter. “Example 1: Ring/parallel journal stream” on page 421 shows a ring configuration with parallel members.

- If the offload stream is a concatenated stream, the current member of the concatenated stream is closed and the next member is left unopened until further offloading begins.
- If any other type of stream is used for the offload stream, the offload stream is left open when the offload process enters an idle state.

Off loading and the CLOSE option

Offloading activity that occurs when the ring is closed depends on the presence or absence of the CLOSE option on the stream definition.

The default action is to fully offload the ring. This implies that the offload stream is opened (if necessary) and all members of the ring containing previously unoffloaded records are scheduled for offloading.

To bypass the default process, specify `CLOSE=NOAUTO`, which implies that the offload stream does not contain all the records that have been written to the ring.

Off loading and system termination

Offloading activity that occurs during system termination is dependent on the option used when the ring is closed.

- The default action, if the offload stream is open when an EOJ command is issued, is to leave the offload stream open even if the offload process enters an idle state.

Leaving the offload stream open allows the contents of the last ring member to be offloaded to the same offload stream member (assuming that the offload stream is a concatenated stream).

- If, prior to the ring stream being closed, the offload process enters an idle state after the EOJ command is issued, the offload stream is closed when the process becomes idle.
- If the option not to have a complete copy of all ring records on the offload stream is taken (`CLOSE=NOAUTO`), any offload process in progress when an EOJ is issued terminates immediately.

I/O error detection

An I/O error is detected for the entire ring when an I/O error occurs on a ring member while it is filling, a ring member while it is offloading, or a control stream on output.

When an I/O error or a file-full condition occurs on the offload stream during output, the ring does not reflect a file-full condition until all remaining empty members are filled.

Off loading ring streams from disk to tape

To off load ring streams from disk to tape, issue one of the following commands:

- `DEFINE STREAM` command with the `AUTOOFFLOAD` option
Members of ring configurations are automatically off loaded when the number of full members reaches the specified threshold.
- `OFFLOAD` command issued by the operator any time during system operation

The ring member being filled when the command is issued is treated as full (at the time of the next write to the ring stream) even if a file-full condition is not reached. Processing is the same as that which occurs automatically when the AUTOOFFLOAD threshold is reached.

Use OFFLOAD in ring configurations to:

- Force offloading if the AUTOOFFLOAD threshold has been set to a high value and system shutdown is anticipated before the value is reached
- Control unplanned requirements of hardware availability, such as tape drives needed for the offload process

If the ring stream already has an active offload process, the command is ignored.

- The COPY command

Issue a COPY command to complete offloading ring streams that are interrupted by system failures or for which the CLOSE=NOAUTO option is specified in the ring stream definition. COPY processing reads the entire stream to the point of the system crash and copies it to another stream.

Use the COPY command if the stream is input to the MERGEJ utility:

- If the original offload stream is defined as a concatenated stream, and an offload was in progress at the time of the crash, the copying process can be started from the beginning of the offload stream member that was in use at the time of the crash.
- If the original offload stream was either a data set or any stream configuration other than concatenated, the entire offload stream must be copied.

Ring streams and recovery

Input ring streams (recovery)

The definition of the input ring stream must be identical to the definition used when the stream was originally created.

When a ring stream is opened for input, the control stream determines the order in which the ring members are processed.

The last active member is normally the first member from which records are retrieved.

To improve recovery performance if recovery input is a ring stream, the recovery routine first attempts to find the specified checkpoint in the ring members before trying to read the offload stream.

I/O error processing for input ring streams

An I/O error is detected for the entire ring when an I/O error occurs on one of the following:

- Offload stream on input
- Control stream on input
- Ring members on input, except for an active member at the time of a system crash

Ring streams as input to RESTART recovery

The following considerations apply when using ring streams as input to RESTART recovery:

- If all the data is migrated to the offload member (through CLOSE=AUTO, for example), you need to specify only the offload member or the input stream. However, this method of recovery is slow.
- If you are uncertain whether all data has migrated to the offload member, define the input stream again as an exact duplicate of the previous output stream.
- During the recovery run, the DEFINE commands must duplicate the definitions in the previous run, with the following exceptions. The DEFINE STREAM name for the previous:
 - Journal stream should be changed from CCAJRNL to CCARF.
 - Checkpoint stream should be changed from CHKPOINT to RESTART.
 - Sub transaction checkpoint stream should be changed from CHPNTS to RESTARTS,
- You must code recovery run DEFINE commands before User 0 input.
- AUTOOFFLOAD is an extraneous keyword during recovery, although it is meaningful in output processing.
- The same data sets used during the run must be used in the recovery run with a DISP of OLD.
- If the CCARF stream is defined as a ring, RESTART recovery attempts to find the roll back checkpoint by first searching the ring members. If the checkpoint is not found, the offload stream is opened from the beginning. You can save substantial recovery time by ensuring that a checkpoint is contained within the ring members.

Concatenated stream configuration

A **concatenated stream** is a group of sequential data sets or streams that are accessed in a serial fashion. I/O is switched to the next member when an end-of-file (EOF) or a stream-full condition is reached.

Concatenated streams reduce the chance of a checkpoint or journal file-full condition.

Processing sequence

Concatenated stream processing follows these steps:

1. All members of the group are opened with a reserve status when a concatenated stream is opened. Open with a reserve status causes Model 204 to:
 - Set up the control blocks necessary to process the stream
 - Enqueue members of the stream to restrict their use as output for other users
 - Perform a recursion check to prevent backward references by any definition in the hierarchy

Open with reserve status does not mean that a physical open was performed or that the allocation was done.

2. If any errors occur during this process, the concatenated stream is considered unusable. Acquired storage is released and an appropriate error message is issued. The first member is physically opened after all members achieve reserve status. If the physical open fails at this point, the concatenated stream is again considered unusable and processing is terminated.
3. After a successful open, records are read from or written to the member until an end-of-data or file-full condition is detected.
4. The next member in the list is physically opened and processed when the end-of-data or file-full condition is recognized.
5. An I/O error that occurs on any member of the stream is reflected immediately as an I/O error for the entire stream.
6. An end-of-data or file-full condition for the concatenated stream is not recognized until the condition occurs on the last member of the stream.

For an example of a concatenated stream, see the section “Example 3: Concatenated tape off load stream for z/OS” on page 426.

Parallel stream configuration

A **parallel stream** is a group of sequential data sets or any combination of streams used in a redundant manner. Every I/O to or from the stream is duplicated on each member, as follows:

- Each logical record written to the stream is duplicated on all the members of the stream.
- Each logical record read from the stream causes a read to be performed on all the members concurrently. The stream read is satisfied by the record from the first member of the stream that completes the input operation without error.

A parallel stream is useful when multiple copies of a data set are required.

Parallel streams and recursions

Model 204 allows only four parallel members and sixteen stream recursion levels. For example, only one member of a two-member parallel stream can have a subsequent parallel stream definition. Likewise, streams may be only sixteen levels deep. If either limit is exceeded, Model 204 displays the following error message:

```
M204.0093: %C IS A RECURSIVE STREAM DEFINITION
```

Checkpoint configuration support

You may define the CHKPOINT and CHPNTS data sets using parallel streams. The only restriction on CHKPOINT and CHPNTS stream definitions is that they may not contain CMS data sets. There are no restrictions on the type of stream you can use to define the CCAJRNL and CCAJLOG journals.

Parallel stream open processing

Parallel stream open processing does the following:

- Allocates storage for control blocks and buffers for all the member streams:
 - Parallel stream open is successful, if the parallel stream control block can be allocated and the number of members successfully opened is equal to or greater than the user-specified minimum (MINAVAIL).
 - Parallel stream open fails, if the stream control block cannot be allocated, or if the user-specified minimum number of members fails to open. Error messages are displayed in the audit trail and at the user's terminal.
- Enqueues physical data sets
- Opens physical data sets

Output parallel streams processing

Output parallel streams are used for checkpoints and journals. An I/O error or file-full condition detected on a member of an output parallel stream causes all the outstanding output operations on all the available stream members to complete.

The following sequence of events occurs:

1. System detects all error conditions at the same point in time.
2. Status of each member of the parallel stream is examined.
3. Number of currently available members that did not have any error conditions is counted.
4. Subsequent error processing depends on two factors:
 - Revised number of available members
 - User-defined minimum available member value (MINAVAIL)

For a description of the MINAVAIL parameter, see the *Rocket Model 204 Parameter and Command Reference*.

Note: Parallel output data sets must all use the same device type and space allocation. Do not specify secondary space.

Output error processing

The following rules apply to output parallel stream error processing:

- If the number of available members with no errors remains at or above the specified MINAVAIL value, the available members with errors are logically removed from the parallel stream and considered unavailable for further output operations. A message M204.0096 is displayed in the audit trail and at the operator console for each unavailable member.

Unavailable members are made available again, if the parallel output stream is rewound.

- If the revised number of available members falls below the specified MINAVAIL value, the parallel processing routine determines the number of records successfully written to each available member stream, regardless of the member's error condition.

Member streams are processed in order by their respective record counts, starting with the member with the fewest records.

- If the number of members available prior to the error was greater than the specified MINAVAIL value, member streams are made unavailable until the number of available members falls below the MINAVAIL value.

When the number of available members falls below the MINAVAIL value, or if the number of members available prior to the error was equal to

MINAVAIL, the following messages are displayed in the audit trail, at the operator console, and at the user's terminal:

```
M204.0095 PARALLEL STREAM stream-name DISABLED. NUMBER
OF MEMBERS BELOW MINAVAIL.
```

```
M204.1826 MEMBER each-member-name OF PARALLEL STREAM
stream-name HAS nnn RECORDS
```

- If any of the selected member streams had either no error conditions or a file-full condition, the parallel stream reflects a file-full condition.
- If all the member streams have I/O error conditions, the parallel stream reflects an I/O error.

Input parallel stream processing

Input parallel streams are used for recovery. When you use the parallel stream for input, you must define the input parallel stream with the same parameters specified in the output definition.

Input stream records are retrieved as follows:

- Records are retrieved simultaneously from all the members, but only one of these members is considered the active member.
- Records retrieved from inactive members are ignored as long as the active member does not encounter an I/O error.
- If the active member encounters an I/O error or an end-of-data condition, the member is closed and the next member in the group is marked as the active member.

The parallel stream input routine uses the same error handling decision logic as output parallel processing, described in the section “Output error processing” on page 418.

Perpetual journaling for z/OS

For z/OS only, Model 204 provides a stream definition of type GDG. A stream defined with the GDG option behaves as a single member ring stream without an offload member. When a GDG generation becomes full, the next generation in sequence is opened. The corresponding generation data set name is written to the stream control file. The control file is written in a readable format so that all members allocated may be inspected.

An example of a CCAJRNL stream for an Online run is:

```
DEFINE STREAM  CCAJRNL WITH SCOPE=SYSTEM GDG=J1 CONTROL=CTLJ
DEFINE DATASET J1 WITH SCOPE=SYSTEM NEW CATALOG -
                DSNNAME=your.ccajrnl.gdgdsn GEN=+1 CYL PRI nnn UNIT=unit
                VOLUME=volser
DEFINE DATASET CTLJ WITH SCOPE=SYSTEM COND CATALOG TRACK PRI 1 -
```

```
DSNAME=your.gdgcontrol.dsn VOLUME=volser UNIT=unit
```

During recovery, the correct GDG generations are automatically opened. When you use a GDG stream for input you must define the GDG input stream with the same parameters specified in the output definition. The corresponding recovery stream would be:

```
DEFINE STREAM CCARF WITH SCOPE=SYSTEM GDG=J1 CONTROL=CTLJ
DEFINE DATASET J1 WITH SCOPE=SYSTEM OLD DSNAME=your.ccajrnl.gdgdsn
DEFINE DATASET CTLJ WITH SCOPE=SYSTEM OLD DSNAME=your.gdgcontrol.dsn
```

Since a GDG data set is dynamically allocated each time it is opened, it must be defined in CCAIN using a DEFINE DATASET command. Otherwise, the Online will terminate during initialization.

Using the GDG option

Caution: *If the SCRATCH option is specified and the LIMIT number for GDGs is exceeded, previous generations are deleted. This can result in a non-recoverable Online, because journal members are missing.*

Rocket Software strongly recommends that you ensure that the LIMIT parameter specified in the GDG base definition is high enough to accommodate all generations required for an Online run. Also, ensure that the default NOSCRATCH option is specified.

Do not allocate GDGs with the SCRATCH option.

Note: The control stream is a very small data set and no DCB attributes are required for its allocation. Also, do not use GDGs as members of ring or concatenated stream for the CHPNTS data set since the close process may lose the ability to determine which GDG generation is being filled.

Data Set Control Blocks (DSCBs) and GDGs

If your site uses pattern DSCBs, you might want to insert the SMSLIKE=*your.site.gdg* option in your DEFINE DATASET command, instead of using the UNIT=*unit* and VOLSER=*volume* options, as shown in the following example.

Set up the output stream using the following code as a guide:

```
DEFINE STREAM CCAJRNL WITH SCOPE=SYSTEM GDG=J1 CONTROL=CTLJ
DEFINE STREAM CHKPOINT WITH SCOPE=SYSTEM GDG=C1 CONTROL=CTLJ
DEFINE DATASET J1 WITH SCOPE=SYSTEM SMSLIKE=pattern.GDG -
    DSNAME=your.site.GDG GEN=+1 CATALOG TRACK PRIMARY=100 -
    NEW LRECL=6749 BLKSIZE=6749 RECFM=F RETRYALLOC=3 -
    RETRYTIME=15
DEFINE DATASET C1 WITH SCOPE=SYSTEM SMSLIKE=pattern.GDG -
```

```
DSNAME=your.site.GDG.CHP GEN=+1 CATALOG TRACK -  
PRIMARY=100 NEW LRECL=6184 BLKSIZE=6184 RECFM=F -  
RETRYALLOC=3 RETRYTIME=15
```

Set up the input stream using the following code as a guide:

```
DEFINE STREAM CCARF WITH SCOPE=SYSTEM GDG=J1 CONTROL=CTLJ  
DEFINE STREAM CHKPOINT WITH SCOPE=SYSTEM GDG=C1 CONTROL=CTLC  
DEFINE DATASET J1 WITH SCOPE=SYSTEM OLD  
DEFINE DATASET C1 WITH SCOPE=SYSTEM OLD
```

Example 1: Ring/parallel journal stream

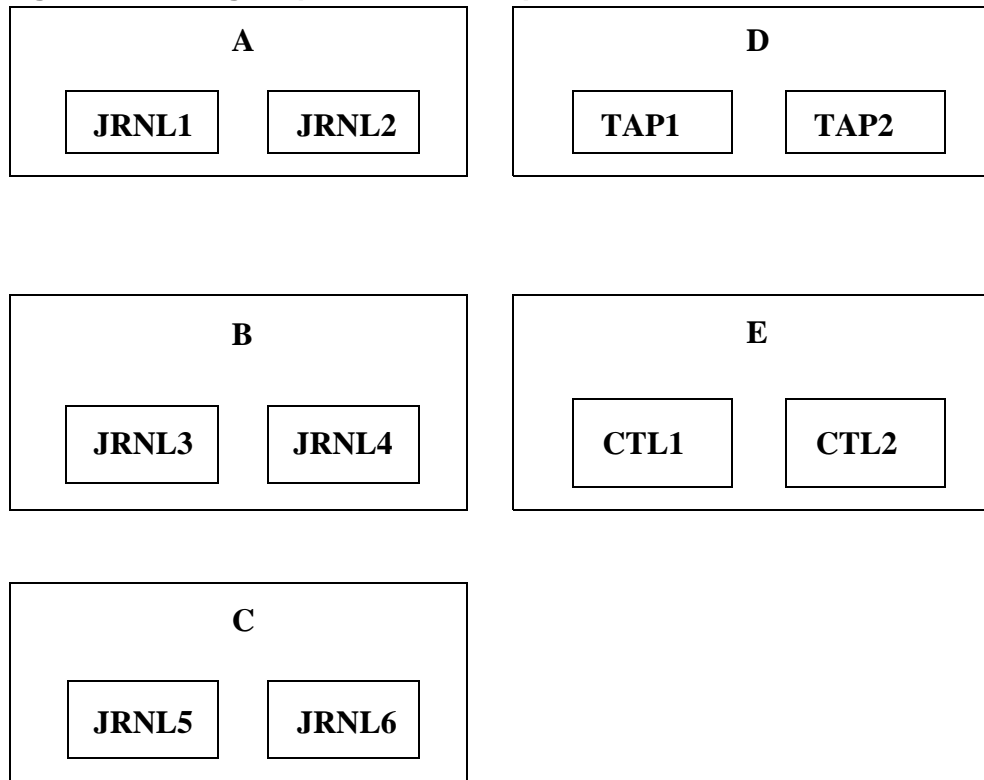
This example illustrates a ring output stream having parallel stream members. The use of a ring configuration reduces the chance of the journal stream becoming full. The use of parallel streams as ring and offload stream members reduces the chance of an I/O error disabling the stream.

In this example:

- OFFLOAD member is a parallel stream made up of two tape data sets.
- Members of the ring output parallel streams are disk streams.
- Defined CCAJRNL stream continues to be a valid output stream as long as at least one of each of the parallel members (MINAVAIL=1) remains available for output.

Figure 17-1 illustrates the concept of a ring output stream with parallel stream members.

Figure 17-1. Ring output stream with parallel stream members



DEFINE STREAM example

The following DEFINE STREAM commands produce the configuration shown in Figure 17-1:

```
DEFINE STREAM CCAJRNL WITH SCOPE=SYSTEM -
    RING=(A,B,C) OFFLOAD=D CONTROL=E AUTOFFLOAD=2
DEFINE STREAM A WITH SCOPE=SYSTEM -
    PARALLEL=(JRNL1,JRNL2) MINAVAIL=1
DEFINE STREAM B WITH SCOPE=SYSTEM -
    PARALLEL=(JRNL3,JRNL4) MINAVAIL=1
DEFINE STREAM C WITH SCOPE=SYSTEM -
    PARALLEL=(JRNL5,JRNL6) MINAVAIL=1
DEFINE STREAM D WITH SCOPE=SYSTEM -
    PARALLEL=(TAP1,TAP2) MINAVAIL=1
DEFINE STREAM E WITH SCOPE=SYSTEM -
    PARALLEL=(CTL1,CTL2) MINAVAIL=1
```

You can use the processing results of this defined CCAJRNL stream as input to recovery.

z/OS JCL

The following example shows z/OS JCL for the ring and parallel journal stream configuration:

```
//JRNL1 DD DSN=M204.JRNL1,DISP=OLD
//JRNL2 DD DSN=M204.JRNL2,DISP=OLD
//JRNL3 DD DSN=M204.JRNL3,DISP=OLD
//JRNL4 DD DSN=M204.JRNL4,DISP=OLD
//JRNL5 DD DSN=M204.JRNL5,DISP=OLD
//JRNL6 DD DSN=M204.JRNL6,DISP=OLD
//CTL1 DD DSN=M204.CTL1,UNIT=DISK,DISP=(NEW,CATLG)
// SPACE=(CYL,(1,1))
//CTL2 DD DSN=M204.CTL2,UNIT=DISK,DISP=(NEW,CATLG)
// SPACE=(CYL,(1,1))
```

Allocate the offload tape data sets dynamically via the DEFINE DATASET command, as in the following z/VSE example.

z/VSE JCL

The following example shows z/VSE JCL for the ring and parallel journal stream configuration:

```
// DLBL JRNL1,'M204.JRNL1',99/365,SD
// EXTENT SYS021,SYSWK1,,,1000,13000
// DLBL JRNL2,'M204.JRNL2',99/365,SD
// EXTENT .....
// DLBL JRNL3,'M204.JRNL3',99/365,SD
// EXTENT .....
// DLBL JRNL4,'M204.JRNL4',99/365,SD
// EXTENT .....
// DLBL JRNL5,'M204.JRNL5',99/365,SD
// EXTENT .....
// DLBL JRNL6,'M204.JRNL6',99/365,SD
// EXTENT .....
// TLBL SYS041,'M204.TAPE1',99/365
// ASSGN SYS040,TAPE
// TLBL SYS040,'M204.TAPE2',99/365
// ASSGN SYS041,TAPE
// DLBL CTL1,'M204.CTL1',99/365,SD
// EXTENT SYS022,SYSWK2,,,1000,100
// DLBL CTL2,'M204.CTL2',99/365,SD
// EXTENT .....
.
.
// EXEC ONLINE,SIZE=AUTO
DEFINE DATASET TAP1 WITH SCOPE=SYSTEM FILENAME=SYS040
DEFINE DATASET TAP2 WITH SCOPE=SYSTEM FILENAME=SYS041
(Other DEFINE statements as shown in the original example)
```

Example 2: Parallel checkpoint stream

```
PAGESZ=6184,RCVOPT=9, . . .
```

CMS FILEDEFS

The following example shows the CMS FILEDEFS for the ring and parallel journal stream configuration:

```
FILEDEF JRNL1 mode DSN M204 JRNL1
FILEDEF JRNL2 mode DSN M204 JRNL2
FILEDEF JRNL3 mode DSN M204 JRNL3
FILEDEF JRNL4 mode DSN M204 JRNL4
FILEDEF JRNL5 mode DSN M204 JRNL5
FILEDEF JRNL6 mode DSN M204 JRNL6
FILEDEF CTL1 mode DSN M204 CTL1
FILEDEF CTL2 mode DSN M204 CTL2
```

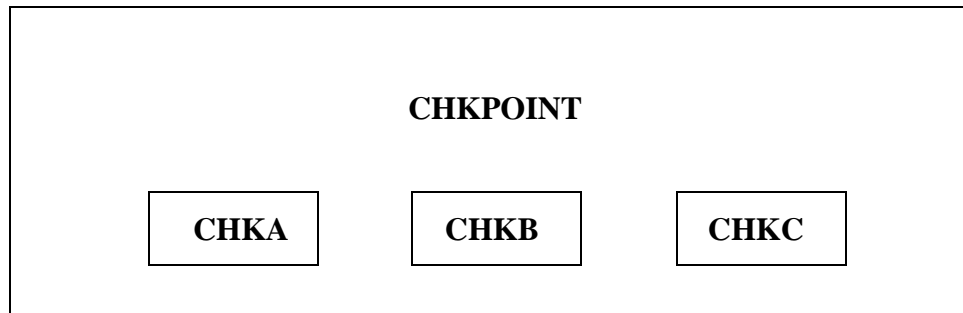
Allocate the offload tape data sets dynamically via the DEFINE DATASET command, as in the previous z/SE example.

Example 2: Parallel checkpoint stream

The following example uses a parallel stream of disk data sets. Because you can use the CPMAX parameter to prevent the checkpoint stream from filling up, define the stream as parallel and not as a ring. The parallel stream also provided duplicate copies of the checkpoint stream.

Conceptually, the checkpoint stream looks like Figure 17-2.

Figure 17-2. Checkpoint stream



DEFINE STREAM command for parallel checkpoint stream

The DEFINE STREAM command used for the parallel stream in Figure 17-2 is:

```
DEFINE STREAM CHKPOINT WITH SCOPE=SYSTEM -
    PARALLEL=(CHKA,CHKB,CHKC) MINAVAIL=2
```

The following considerations apply to this DEFINE STREAM command:

- Only one DEFINE STREAM command is required for this configuration of a checkpoint stream.

- To guarantee that you always have at least two good copies, set MINAVAIL to 2.

The CHKPOINT stream continues to be valid output as long as at least two of the parallel members remain available for output.

Recovery processing

For a recovery run, the following User 0 input line is required:

```
DEFINE STREAM RESTART WITH SCOPE=SYSTEM -  
    PARALLEL=(CHKA,CHKB,CHKC) MINAVAIL=2
```

Note: Only the STREAM option of the DEFINE command is valid for a recovery processing.

z/OS JCL

The following example shows the z/OS JCL for a parallel checkpoint stream configuration:

```
// CHKA DD DSN=M204.CHKA,DISP=OLD  
// CHKB DD DSN=M204.CHKB,DISP=OLD  
// CHKC DD DSN=M204.CHKC,DISP=OLD
```

z/VSE JCL

The following example shows the z/VSE JCL for a parallel checkpoint stream configuration:

- Original run

```
// DLBL CHKA,'M204.CHKA',,99/365,SD  
// EXTENT SYS021,SYSWK1,,,1000,13000  
// DLBL CHKB,'M204.CHKB',,99/365,SD  
// EXTENT SYS021,SYSWK1,,,1400,13000  
// DLBL CHKC,'M204.CHKC',,99/365,SD  
// EXTENT SYS021,SYSWK1,,,2700,13000
```

- Recovery run

```
// DLBL CHKA,'M204.CHKA',,DA  
// EXTENT SYS021,SYSWK1,,,1000,13000  
// DLBL CHKB,'M204.CHKB',,DA  
// EXTENT SYS021,SYSWK1,,,1400,13000  
// DLBL CHKC,'M204.CHKC',,DA  
// EXTENT SYS021,SYSWK1,,,2700,13000
```

CMS FILEDEFs

The following example shows the CMS FILEDEFs for a parallel checkpoint stream configuration:

```
FILEDEF CHKA mode DSN M204 CHKA
FILEDEF CHKB mode DSN M204 CHKB
FILEDEF CHKC mode DSN M204 CHKC
```

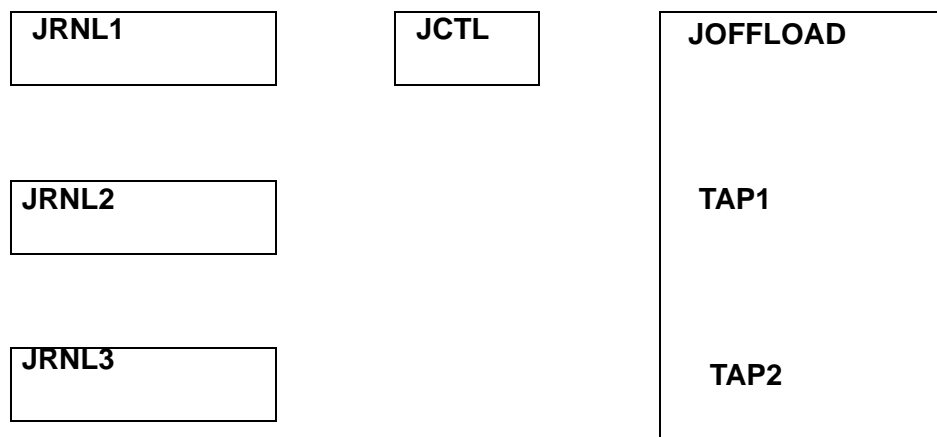
Example 3: Concatenated tape off load stream for z/OS

If tape drives are limited and you cannot dedicate one drive for a tape offload stream, you can define the offload stream as a concatenated stream. When the offload stream is concatenated, each member is physically opened only when an offload is required and is automatically closed when the offload is completed.

With the use of dynamic allocation during open, a DEFINE DATASET command for each member of the concatenated offload stream is sufficient to ensure that tape drives are allocated during offload processing and freed when the offload completes. When the offload process is idle, tape drives are not necessary. No DD card referencing the offload stream should be present.

Conceptually, the stream configuration looks like Figure 17-3.

Figure 17-3. Stream configuration



DEFINE STREAM commands

The DEFINE STREAM commands used for the concatenated stream shown in Figure 17-3 are:

```
DEFINE STREAM CCAJRNL WITH SCOPE=SYSTEM -
    RING=(JRNL1,JRNL2,JRNL3) OFFLOAD=JOFFLOAD -
    CONTROL=JCTL CLOSE=NOAUTO AUTOOFFLOAD=2
DEFINE STREAM JOFFLOAD WITH SCOPE=SYSTEM -
    CONCATENATE=(TAP1,TAP2)
```

```
DEFINE DATASET TAP1 WITH SCOPE=SYSTEM          -  
    DSN=M204.TAPE1 UNIT=TAPE CATALOG  
DEFINE DATASET TAP2 WITH SCOPE=SYSTEM          -  
    DSN=M204.TAPE2 UNIT=TAPE CATALOG
```

The following considerations apply to this DEFINE STREAM command:

- The AUTOOFFLOAD value of 2 triggers the offload process when ring members JRNL1 and JRNL2 fill up.
At this point, the concatenated offload member TAP1 is physically opened and a tape mount is requested.
- When the offload of JRNL1 and JRNL2 completes, TAP1 is closed and the tape drive is freed. Other tape mounts are not requested until the offload process is once more triggered, this time when JRNL3 and JRNL1 fill up.
- If JRNL3 does not fill up before termination processing, other offloads cannot be done, because the DEFINE STREAM command specified CLOSE=NOAUTO.

JCL for concatenated offload

The JCL required for the ring members and the control data set is:

```
//JRNL1 DD DSN=M204.JRNL1,DISP=OLD  
//JRNL2 DD DSN=M204.JRNL2,DISP=OLD  
//JRNL3 DD DSN=M204.JRNL3,DISP=OLD  
//JCTL DD DSN=M204.JCTL,DISP=OLD
```

Troubleshooting GDG streams using specific generations

In the unfortunate event that when using GDG streams, you need to use specific generations for recovery, take the following steps in the following example:

1. Your Online job using a journal GDG stream crashes. It creates journals G0001V00 and G0002V00.
2. You incorrectly submit the Online again without running recovery. This run creates journal G0003V00.
3. You bring down the second Online and want to run recovery for the first Online. Note that only files that were not updated in the second run can be recovered via RESTART recovery. Files that were updated in the second run would require media recovery.

At this point, you can no longer use the GDG stream definition for CCARF because the GDG stream control data set now points to G0003V00 as the current generation.

To recover from the initial crash, you must determine all of the generations used in that job and create single input data sets for CCARF and RESTART (if you used a GDG stream for checkpointing).

For example, if the run that crashed created generations 1 and 2 for the journal and 97 through 100 for the checkpoint, you could create single data sets with the following JCL:

```
//CCARF      EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//* combine all journal generation data sets into one file:
//SYSUT1     DD DISP=SHR,DSN=xxx.CCAJRNL.G0001V00
//           DD DISP=SHR,DSN=xxx.CCAJRNL.G0002V00
//SYSUT2     DD DISP=SHR,DSN=xxx.single.CCARF
//SYSIN DD DUMMY
//*
//RESTART    EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//* combine checkpoint generation data sets into one file:
//SYSUT1     DD DISP=SHR,DSN=xxx.CHKPOINT.G0097V00
//           DD DISP=SHR,DSN=xxx.CHKPOINT.G0097V00
//           DD DISP=SHR,DSN=xxx.CHKPOINT.G0098V00
//           DD DISP=SHR,DSN=xxx.CHKPOINT.G0099V00
//           DD DISP=SHR,DSN=xxx.CHKPOINT.G0100V00
//SYSUT2     DD DISP=SHR,DSN=xxx.single.RESTART
//SYSIN DD DUMMY
```

The single data sets can then be used in your recovery job. If you need to use specific generations for media recovery, special considerations apply when using GDG streams.

Part V

Performance



This part describes the Model 204 system manager's role in troubleshooting.

18

Performance Monitoring and Tuning

In this chapter

- Overview
- Optimizing operating system resources
- Timer PC
- EXCPVR
- IOS BRANCH ENTRY
- Disk buffer monitor statistics and parameters
- Page fixing
- 1MB virtual pages
- Offloading Model 204 work to zIIP processors
- Balancing channel load
- Reducing storage requirements
- Dynamic storage allocation tracking
- Cache fast write for CCATEMP and CCASERV
- Sequential processing
- Long requests

- IFAM2 control
- SQL system and user statistics
- z/OS recovery
- Resident Request feature for precompiled procedures
- Multiprocessing (MP/204)
- MP performance and tuning
- MP User Language statement processing
- MP program constructs
- Optimization using MPOPTS and MP OPTIONS

Overview

This chapter discusses several methods of enhancing Model 204 system performance.

The last two sections of this chapter discuss MP/204 (multiprocessing), a Rocket Software product available to z/OS users with multiple processor hardware configurations. Multiprocessing significantly increases throughput for most Model 204 applications.

Optimizing operating system resources

Because Model 204 runs as a multiuser, interactive task, take care at the operating system level (z/OS, z/VM, z/VSE) to ensure that Model 204 has adequate access to system resources.

The resources most critical to Model 204 performance are CPU and real memory or central storage. Because Model 204 supports many users from one virtual storage space, treat it as a high priority, system-level task. Address the following issues at your installation.

z/OS

Run Model 204 as a nonswappable address space (see the *Rocket Model 204 z/OS Installation Manual* for details).

Run Model 204 in a performance group with a high, fixed dispatching priority rather than mean-time-to-wait or rotate (supported only in z/OS releases prior to z/OS 2.2).

Limit the paging rate of the Model 204 address space to no more than one to two pages per second through storage isolation or working-set-size protection. These controls are provided through the System Resource Manager (SRM) component of z/OS via the parameters PWSS and PPGRT.

z/VM

Issue the following commands for the Model 204 service machine:

For z/VM/ESA:

```
SET QUICKDSP service_machine_name ON
SET RESERVED service_machine_name nnn
```

where *nnn* = 70-80% of STORINIT.

z/VSE

Run Model 204 in a high priority partition.

Ensure that sufficient real memory is available to support the partition size defined and to guarantee a paging rate not to exceed one to two pages per second.

Timer PC

Timer PC is a Cross-Memory Services Facility option that provides a problem-program interface to the job step CPU accounting information maintained by SMF that is functionally equivalent to TTIMER/STIMER processing, but consumes less CPU time. The saving is most visible in environments where the user switching rate is very high: for example, when a buffer pool is heavily loaded and every page request will include a wait.

Timer PC requires installation of only the Cross-Memory SVC. The Cross-Memory Services Facility, which establishes the PC environment during Model 204 initialization, is used by Timer PC to avoid SVC interruptions.

To enable Timer PC, specify the XMEMOPT and XMEMSVC parameters on User 0's parameter line.

- To enable Timer PC, the XMEMOPT setting must include the X'01' bit.
- To disable Timer PC, the XMEMOPT setting must not include the X'01' bit.

For a description of the possible settings of XMEMOPT, see the *Rocket Model 204 Parameter and Command Reference*.

EXCPVR

Under z/OS systems that are not XA or ESA, the EXCPVR feature provides CPU savings when performing disk and server I/O. EXCPVR is not supported for z/OS in 64-bit mode.

Using EXCPVR, Model 204 translates its own CCW data addresses from virtual to real.

Under z/OS, disable EXCPVR (EXCPVR=0) in favor of IOS BRANCH ENTRY, a feature discussed in the section “IOS BRANCH ENTRY” on page 434. EXCPVR and IOS BRANCH ENTRY are mutually exclusive.

Under z/OS, ECKD channel support (discussed in “ECKD channel program support” on page 170) is not available for use with EXCPVR. Do not use 9345 DASD with EXCPVR because of the potential for poor performance on writes.

The following considerations apply.

To enable EXCPVR:

- Install the Model 204 page fix/start I/O appendage in SYS1.SVCLIB (see the appropriate Rocket Model 204 Installation Guide).
- Authorize the version of Model 204 that uses EXCPVR under the Authorized Program Facility (APF).

Authorization procedures differ among VS systems (see the appropriate Rocket Model 204 Installation Guide for systems that support EXCPVR).

To use EXCPVR:

- Set the EXCPVR parameter to the last two characters of the name of the PGFX/SIO appendage installed in SYS1.SVCLIB.
- The suggested value of the EXCPVR parameter is C 'W2', but you can use other characters between C 'WA' and C 'Z9'. The default is binary zero (EXCPVR not used).
- You can set EXCPVR in the EXEC statement or on User 0's parameter line.

IOS BRANCH ENTRY

IOS BRANCH ENTRY is a Cross-Memory Services facility option that provides high-performance I/O for Model 204 running under z/OS.

IOS BRANCH ENTRY provides:

- Fast path to the I/O Supervisor (IOS)
- Reduced instruction path length for each I/O operation by translating a pre-formatted channel program
- Page fixing for required storage areas
- Direct branching to IOS to initiate I/O activity
- Placement of all database I/O control blocks, hash cells, page fix lists, and buffers above the 16-megabyte line, significantly reducing the risk of control block overflow into CSA

IOS BRANCH ENTRY replaces the EXCP driver and overrides the EXCPVR option if both EXCPVR and IOS BRANCH ENTRY are requested.

Activating IOS BRANCH ENTRY

To enable IOS BRANCH ENTRY, perform the following steps:

1. Install the Model 204 Cross-Memory Services SVC.
2. Authorize Model 204 under APF.
3. Specify XMEMOPT and XMEMSVC parameters on the User 0 input line. See the *Rocket Model 204 Parameter and Command Reference* for appropriate settings:
 - XMEMOPT: Cross-memory services options
 - SMEMSVC: Number of cross-memory services SVC

When IOS BRANCH ENTRY is enabled, control blocks, hash cells, and page fix lists are allocated above the 16-megabyte line.

Saving storage when using IOS Branch Entry

During initialization Model 204 calculates a default number of Disk Buffer I/O control blocks (DBIDs). If you are using IOS Branch Entry, this default number may be unnecessarily high. Use the MAXSIMIO parameter to allocate a smaller number of DBIDs. (See “MAXSIMIO: Maximum number of simultaneous disk I/Os” in *Model 204 Parameter and Command Reference*.)

Model 204 uses the following formula to calculate, as a default, the maximum number of DBIDs that will ever be needed:

$$32 * (NSUBTKS + NSERVS)$$

Using a lower MAXSIMIO value when you are also using IOS Branch Entry limits the number of DBIDs allocated and results in storage savings. If you are not using IOS Branch Entry, an explicit MAXSIMIO setting is ignored and the value calculated by Model 204 is used instead.

Disk buffer monitor statistics and parameters

DKBM efficiency

The efficiency of the buffering monitor in keeping needed disk pages in memory is measured by the DKPR, DKRD, and DKRR statistics.

DKRD and DKPR statistics

DKRD maintains a count of the number of real page reads that occur from Model 204 files. DKPR measures the number of times a user requests a page to be opened.

A real read, which increments DKRD, is required only if the page requested is not already contained in a buffer. Pages cannot be kept open across any function that causes a real-time break, such as terminal I/O.

If the value of DKRD is close to the value of DKPR and many pages are bumped out of buffers during user waits, increase the number of buffers in the run.

DKRR statistic

DKRR is a more accurate measure of the efficiency of the buffering monitor in retaining previously used disk pages in memory.

DKRR counts the number of recently referenced pages that must be reread from disk. A list of each user's recently used pages is kept. If a new page reference is made and the page is already in the table, DKRR is incremented.

New pages are added at the end of the table. If a page already appears in the table, the older entry is deleted and the table is maintained in page-reference order. If a new entry does not fit in the table, the oldest entry is deleted. A new reference to the deleted page does not increment DKRR.

DKRR is controlled by:

Parameter	Means	Usage
LRUPG	Number of page IDs retained for each user for matching.	You must set LRUPG in the parameter statement for the operating system, with a value between 0 and 255. The value 0 indicates that no DKRR statistics are kept.
LRUTIM	Controls the aging of the IDs in the table and is set on User 0's parameter line.	If a user does not make a disk reference within LRUTIM milliseconds of the previous disk reference, all the disk page IDs previously remembered are considered obsolete and deleted.

Adequacy of existing buffers

Adequacy of existing buffers is monitored by the FBWT, DKSFBFS, DKSAWx, DKSDIR, DKSDIRT, DKSKIP, DKSRR, DKSTBLx, DKSWRP, and DKSWRPT statistics:

- DKSFBFS records how many times a scan of the queue of free buffers is required when the oldest free buffer is not immediately available. The oldest buffer can be either in the process of being written out, or waiting to be written because its contents have been modified.

- DKSRR measures the number of times an expected buffer page requires I/O to be read.

Buffers currently open are not in the reuse queue. To avoid the need for real I/O, requests for pages found in LRU (reuse) queue buffers cause the buffers to be removed from the queue. The oldest buffer that has not been updated (and does not need to be written) is the first to be reused.

Model 204 remembers modified buffers that were skipped over in a buffer scan. In anticipation of new buffer requests, some of the remembered buffers might be written out so that the buffers can be reused:

- DKSKIP is the high-water mark of the number of buffers skipped while searching for one that could be reused immediately.
- DKSKIPT is the total number of skipped buffers for the run.

Anticipatory writes

An anticipatory write is the write of a closed, modified page from the buffer pool back to CCATEMP or to the file it came from in anticipation of the need for a free buffer. Closed pages in the buffer pool are arranged in the LRU (least recently used) queue from least recently referenced (bottom of queue) to most recently referenced (top of queue). When a page is closed, meaning no user is referencing it, the buffer containing that page is usually placed at the top of the LRU queue. Over time, buffers, and the pages they contain, migrate to the bottom of the LRU queue.

Unmodified pages will not be written since no changes have occurred. However, the buffer occupied by that page is immediately available for reuse when the buffer reaches the bottom of the LRU queue. The same is true for modified pages except that they must first be written before the occupied buffer can be reused. Pages that are on the LRU queue and are subsequently referenced are removed from the LRU queue and are not available for reuse until again closed by all users.

Understanding the anticipatory write window

The anticipatory write window is a collection of buffers at the bottom of the LRU queue. The window starts at LDKBMWND buffers from the bottom of the queue. When a modified page migrates into the window, the write for the page is started. The larger the value of LDKBMWND, the sooner that anticipatory write starts.

The purpose of this window is to provide a mechanism that guarantees that the buffers at the bottom of the reuse queue are immediately available for reuse. Buffers containing unmodified pages are immediately available. Buffers containing modified pages, however, are not immediately available unless the write for the page has completed.

Monitoring statistics for LDKBMWND

Setting LDKBMWND to a high value, say more than one half NUMBUF, may cause an undesirable increase in DKWR. Setting LDKBMWND to a low value, say NSERVS+1 (the default) in a heavy update system, may cause users to wait for buffers to become reusable and cause the FBWT statistic to increase. Start with LDKBMWND equal to $.25 * \text{NUMBUF}$ and monitor the following statistics:

- DKSAWW
- DKSKIP
- DKSKIPT
- DKWR
- DKWRL
- FBWT

If you have buffers allocated above the bar, the same window is created for those buffers and its size is controlled via LDKBMWNG. If you use MP/204, then you may have multiple LRU queues; the number is set with the NLRUQ parameter.

The system manager can adjust the size of the anticipatory write window by resetting the LDKBMWND parameter. The default size of LDKBMWND is $\text{NSERVS} + 1$, with a minimum size of 3.

- High values might unnecessarily increase the number of writes (measured by the DKWR statistic).
- Low values might cause excessive waiting for buffers (measured by the FBWT statistic).
- DKSAWx statistics measure the number of anticipatory writes done under the following conditions:

Statistic	When a buffer was...
DKSAWB	Written from the buffer pool. The page in the buffer is then deleted and no longer available without a re-read. Usually a small statistic and typically only incremented by the INITIALIZE command and in a few other, rare events. This is not an anticipatory write.
DKSAWW	The number of anticipatory writes of buffers that entered the anticipatory write window.

- FBWT measures the number of times a requested buffer could not be obtained until a write was complete. If this number is high, it may be appropriate to increase the value of LDKBMWND.
- DKSTBLA, DKSTBLB, DKSTBLC, DKSTBLD, and DKSTBLF counters measure how real I/O to read pages of database files breaks down by table.

- DKSWRP and DKSWRPT, which measure the high-water mark and total number of writes in process within a monitoring window, and DKSDIR and DKSDIRT, which measure the high-water mark and total number of modified buffers within the window, are controlled by the LDKBMW parameter. LDKBMW specifies the number of buffers examined by the Disk Buffer Monitor at each find buffer operation.

The monitoring window starts at the bottom of the reuse queue, and extends upward for the specified number of buffers. If LDKBMW is nonzero, every call to find a buffer causes Model 204 to examine the window. Because the overhead associated with this parameter is significant, Rocket Software recommends that it remain at its default value of zero, unless directed otherwise by a Rocket Software Technical Support engineer.

A system manager can reset the LDKBMW parameter during a run.

A complete description of the LDKBMW and LDKBMWND parameters is given in the *Model 204 Parameter and Command Reference*.

Disk update file statistics

The Model 204 disk update file statistics are displayed when the last user closes the file or when partial file statistics are gathered.

Statistic	Records elapsed time in milliseconds...	Comments
DKUPTIME	To write a file's pages to disk and to mark it "physically consistent" on disk.	Includes all time spent writing pages, even if the disk update process was interrupted, as indicated by the following message: M204.0440: FILE <i>filename</i> DISK UPDATE ABORTED
PNDGTIME	File waits to be written to disk, after the last update unit completed.	The measured interval begins at the completion of the last update unit for the file, and ends when a user (or possibly the CHKPPST) begins to write the file's modified pages to disk. PNDGTIME statistic is accumulated only when the DKUPDTWT parameter is nonzero. Model 204 does not display or audit statistics with a value of zero.
UPDTTIME	Open file is part of at least one update unit.	

Page fixing

Use page fixing to reduce paging traffic in z/OS systems by using the z/OS PGFIX macro to fix heavily used sections of Model 204 in memory. You must

authorize Model 204 under the VS Authorized Program Facility (APF) if any pages are to be fixed.

You can fix the following areas in memory by specifying a summed setting on the PAGEFIX parameter on User 0's parameter line:

- KOMM
- Scheduler work areas
- Server swapping work areas
- Statistics area
- Buffer hash field
- Buffer control blocks
- Buffers
- Journal control blocks and buffers
- Physical extent blocks for user files
- Physical extent blocks for CCATEMP
- LPM and DSL for user files
- LPM and DSL for CCATEMP
- DCB areas
- File save areas
- User/file mode table
- Resource enqueueing table
- Record enqueueing table
- Server areas
- Record descriptions
- Performance subtask work area

To prevent the operating system from paging in buffers only to have the data replaced, memory pages containing Model 204 file buffers are released and reallocated before disk read operations. Areas can vary in size, depending on the settings of various parameters such as NUSERS, NFILES, and MAXBUF.

Fixing too much in real memory might increase the operating system paging rate and adversely affect the performance of other z/OS tasks.

The use of this parameter is generally *not* recommended if the paging rate of the Model 204 address space is controlled through working-set size protection (discussed on “Optimizing operating system resources” on page 432).

1MB virtual pages

Model 204 supports 1MB virtual pages in this environment:

- CPU: on model z/10 and later
- O/S: z/OS 1.9 and later

1MB pages can be used only with storage above the bar (2GB).

Use of 1MB pages provides a performance advantage because the operating system can now keep more real addresses in memory and minimize the translation of virtual to real addresses.

To use 1MB pages, set the Model 204 parameter, ZPAGEOPT, to indicate which areas should be allocated using 1MB pages.

The size of the virtual storage area with 1MB pages for an LPAR is defined at IPL time by the z/OS LFAREA (Large Frame AREA) parameter, set in SYS1.PARMLIB(IEASYSnn). This storage is not pageable.

Please consult your systems programmer to:

- check on the setting of the LFAREA parameter, and
- confirm that your operating system has all the latest maintenance from IBM regarding 1MB pages.

The LFAREA is dynamic and shared by all regions. When z/OS experiences a shortage of 4K pages, it steals 1MB pages and subdivides them into 4K pages. If the Model 204 request for virtual storage with 1MB pages cannot be satisfied, then regular 4K pages will be used for this purpose, without generating a warning.

When buffers are allocated with 1MB pages, Model 204 must use IOS Branch Entry (XMEMOPT=2). EXCP and EXCPVR do not support buffers allocated with 1MB pages above the bar.

Parameters

The ZPAGEOPT parameter indicates which areas should be allocated using 1MB pages.

Two view-only parameters provide information on 1MB pages:

- STORZPAG - CURRENT MBYTES STORAGE ALLOCATED WITH 1 MB PAGES
The current number of megabytes of above the bar storage allocated with 1MB pages.
- STORMXZP - HWM MBYTES STORAGE ALLOCATED WITH 1 MB PAGES

The high water mark of megabytes of above the bar storage allocated with 1MB pages during the run.

Options The MONITOR GSTORAGE option displays statistics regarding allocated above the bar storage.

Messages The following messages are associated with 1MB pages and above the bar storage:

- M204.2919:HWM MBYTES ATB STORAGE ALLOCATED = %C
- M204.2925:HWM MBYTES STORAGE ALLOCATED WITH 1MB PAGES = %C

MONITOR GSTORAGE The MONITOR GSTORAGE (monitor grande storage) option displays statistics regarding allocated storage and 1MB pages. It can be abbreviated as MONITOR GS or M GS.

MONITOR GSTORAGE uses the following format:

```
NAME START SIZE (MB) 1M SIZE PGFIX
```

where,

- NAME - name of the memory object.
Buffers are allocated in chunks and presented in one line using name "BUFFERS" and accumulated data for all chunks.
- START - starting address of the memory object. For buffers it is the address of the first chunk.
- SIZE (MB) - Size in MB of the memory object. For buffers it is the accumulated size of all chunks.
- 1M SIZE - Size in MB of memory object backed by z/OS 1MB pages. For buffers some chunks may be backed by 1MB pages and some by 4K pages, so values in columns SIZE (MB) and 1M SIZE may be different.
- PGFIX - a letter "Y" in this column indicates the storage is page fixed.

There are three statistical messages printed below the table, indicating total amount of storage allocated above the bar, total amount of storage allocated with 1MB pages, and total amount of page fixed storage. All amounts are in MB. Data presented by the MONITOR GSTORAGE represent current values.

Example

The following example shows output from a MONITOR G STORAGE command, showing the amount of storage that could – and could not – be allocated above the bar and in 1MB pages:

```

11.238  AUG 26  14.30.43          PAGE 1
NAME           START           SIZE (MB)  1M SIZE  PGFIX
RECLTBL      00000048_00100000           1         1      Y
CCASERV      00000048_00C00000          10        10
CCATEMP      00000048_08C00000         118
CCATDBC      00000048_10300000           1         1
BUFFHASH     00000048_10500000           1         1      Y
BUFFCNTL     00000048_10A00000           4         4
CCAAPSY      00000048_1FE00000           2
BUFFERS      00000048_11200000         119        72
Total amount of storage allocated above the bar:      256
Total amount of storage backed by 1 MB pages   :      89
Total amount of storage page fixed             :         2

```

Offloading Model 204 work to zIIP processors

On a z/OS system, Model 204 zIIP support enables you to offload Model 204 work from regular processors to zIIP processors.

Model 204 can run a portion of workload on IBM System z Integrated Information Processor (zIIP) engines. The only unit of work that z/OS schedules to run on a zIIP engine is the enclave Service Request Block (SRB). An enclave is a collection of resources, such as address spaces, TCBs, and SRBs, that are controlled by Workload Manager (WLM) as an entity.

Model 204 creates enclave SRBs to do work on a zIIP processor. These SRBs are managed in the same way as MP subtasks. Therefore, the term *zIIP subtask* is used to refer to Model 204 enclave SRBs.

Note: MP subtasks and zIIP subtasks may be used together. Therefore, AMPSUBS, NMPSUBS, AMPSUBZ and NMPSUBZ may all be greater than zero. However, to avoid unnecessary overhead:

- Do not set AMPSUBS greater than the number of central processors actually available.
- Do not set AMPSUBZ greater than the number of zIIP processors actually available.

Model 204 workload eligible for zIIP offload

Components of Model 204 that are eligible for zIIP processing include:

- FIND statement for files and groups
- SORT statement, when there are more than 64 records
- APSY load

- Server swapping
- FOR loops, if they do not:
 - perform updates
 - issue READ SCREEN statements
 - issue MQ calls (for example, MQGET)
 - issue calls to mathematical functions (for example, \$ARC)
 - read or write external, sequential files

Model 204 parameters for zIIP support

The following parameters, modeled from their MP counterparts, are provided for zIIP support:

- AMPSUBZ: Specifies the number of active zIIP subtasks
- NMPSUBZ: Specifies the number of zIIP subtasks attached during initialization (enclave SRBs)
- SCHDOFLZ: Specifies the target number of threads on the zIIP offload queue per active zIIP subtask
- SCHDOPT1: Allows server swapping by an MP subtask when all zIIP subtasks are busy
- ZQMAX: Limits the number of work units for execution by the zIIP subtask

Additionally, values in the SCHDOPT parameter support zIIP processing, as described below.

SCHDOPT

The following SCHDOPT values are related to zIIP support:

- X'08': Do server I/O on MP subtask.
- X'20': Do not let the maintask pick up zIIP workload, even if no other work is available.
- X'40': Do not let an MP subtask pick up zIIP workload, even if no other work is available.
- X'80': Allow zIIP to do server swapping. If you have SCHDOPT=X'80' set and you also use the MP/204 feature, then you might want to set SCHDOPT1=X'01' as described in the "Usage Notes" below.

The SCHDOPT parameter setting of X'80' allows server swapping to be offloaded to zIIP processors.

The X'80' setting is only valid when CCASERVER is in memory (servers swapped into memory). If you set SCHDOPT=X'80' when CCASERVER is not in memory, the setting X'80' is reset and message 2914 is issued:

SCHDOPT INDICATION OF SERVER SWAPPING DONE BY zIIP IS ONLY VALID WHEN CCASERV IS IN MEMORY

Initialization then continues.

- Usage Notes**
- SCHDOPT=X'08': An MP subtask will always execute a swap unit of work before a user unit of work. This could mean that every MP subtask is busy swapping.
 - SCHDOPT=X'80': zIIP subtasks are allowed to perform server swapping.
 - SCHDOPT=X'88': The zIIP subtask is behaving exactly as an MP subtask. Both zIIP and MP subtasks could be busy doing swapping while no work is actually being done.
 - SCHDOPT=X'80' and SCHDOPT1=X'01': The zIIP subtask will do the swapping before doing any work and the MP subtasks will do real user work. But if there is no user work to do, the MP subtask will help the zIIP subtask by executing a swap unit of work. That way, if the zIIP subtask gets behind on its swapping, then the MP subtask can assist.

SCHDOPT1 The parameter SCHDOPT1 uses an x'01' value. SCHDOPT1=x'01' allows an MP subtask to handle server swapping if the subtask has nothing else to do.

SCHDOPT1 is useful when zIIP is handling server swapping and needs help from an MP subtask.

Limiting the number of work units for execution by the zIIP subtask The ZQMAX parameter limits the number of work units that may be placed for execution by the zIIP subtask.

The default and maximum value for ZQMAX is 65535.

- If ZQMAX is set to a non-zero value:
When a work unit is pushed onto the zIIP queue and the current zIIP queue length exceeds the value of ZQMAX, then that work unit is placed on the MP stack instead. Effectively, this limits the amount of work that is done on the zIIP and prevents zIIP overload.
- If ZQMAX is set to zero:
No users will be added to the zIIP queue. However, if SCHDOPT-X'80' is on and a zIIP subtask is active (AMPSUBZ>0), then server swapping will occur on the zIIP subtask without the overhead of pushing zIIP work units onto the zIIP queue.

Model 204 zIIP infrastructure created during initialization If NMPSUBZ is greater than zero, then that number of zIIP subtasks is created at initialization. Each zIIP subtask is an enclave SRB.

AMPSUBZ specifies the number of active zIIP subtasks and can be reset during the run.

Tuning the zIIP subtask activation scheduler The SCHDOFLZ parameter specifies the target number of threads on the zIIP offload queue per active zIIP subtask. Setting this parameter makes the Model 204 zIIP subtask scheduler more responsive to instantaneous increases in load.

The default value of 2 means that if there are more than twice as many units of zIIP offloadable work waiting to be processed in a zIIP subtask as there are active subtasks, and fewer than AMPSUBZ subtasks are currently running, another zIIP offload subtask (not to exceed AMPSUBZ) is activated. Subtasks are deactivated, but remain available, when they have no work to do. This deactivation helps minimize zIIP overhead.

The default value is recommended in most cases.

- Setting the value to 1 potentially increases throughput but also increases overhead, which is not recommended unless your site has plenty of spare CPU capacity.
- A value of 0 is not recommended because it will likely result in the unnecessary activation of zIIP subtasks -- by the time some of them are dispatched, other zIIP subtasks will have handled all pending units of work.
- A value greater than 2 might decrease CPU overhead while reducing throughput by reducing parallelism.

Model 204 zIIP-related statistics and monitoring The MONITOR TASKS command indicates a zIIP subtask by displaying a lowercase letter z beside the subtask number. The following are zIIP system statistics: ZDEQ, SDEQ, and ZIPCPU. See “zIIP statistics” on page 446 for details.

zIIP utilization zIIP utilization can be regulated with the IIPHONORPRIORITY parameter.

The z/OS parameter IIPHONORPRIORITY, which defaults to YES, honors the existing task processing priority. This means that a zIIP engine executes workload only when all regular engines are busy.

If zIIP usage is more important to you than regular CPU usage, then you should set IIPHONORPRIORITY=NO and the Model 204 SCHDOPT parameter should include X'20' and X'40' values.

zIIP-compatible functions All Rocket Software supplied User Language \$functions are zIIP compatible.

zIIP statistics The MONITOR STAT command output will display the zIIP statistics described in Table 18-1.

Table 18-1. MONITOR STAT command zIIP and related statistics

Statistic	Total CPU time in milliseconds used by...
CPUTOTZE	workload eligible for offloading to a zIIP engine

Table 18-1. MONITOR STAT command zIIP and related statistics

Statistic	Total CPU time in milliseconds used by...
CPUONZIP	all zIIP engines
Statistic	Number of times that a unit of work was taken by...
MTDEQ	The MAINTASK from the MAINTASK queue
MTSDEQ	The MAINTASK from an MP subtask queue
MTZDEQ	The MAINTASK from a zIIP subtask queue
STDEQ	An MP subtask from an MP subtask queue
STZDEQ	An MP subtask from a zIIP subtask queue
ZTDEQ	A zIIP subtask from a zIIP subtask queue

MTZDEQ and STZDEQ show the number of times that the maintask or an MP subtask took work that was allocated to a zIIP subtask.

The MAINTASK can take work from an MP subtask queue or a zIIP subtask queue. An MP subtask can take work from a zIIP subtask queue. A zIIP subtask can only take work from a zIIP subtask queue.

To see non-zero results for MTZDEQ or STZDEQ, you must turn off SCHDOPT x'20' and SCHDOPT x'40'. If SCHDOPT x'20' is on then the maintask is not allowed to take zIIP work. Likewise, SCHDOPT x'40' prevents the MP subtask from taking from zIIP. So, if either bit is on, the corresponding statistic will always be zero.

A MONITOR TASKS command displays three zIIP columns visible on screens formatted as a 3270 Model 5 terminal.

The columns are:

- SDEQ, which represents the number of units of work taken from an MP subtask queue by the maintask. This column has meaning only for the maintask; for other tasks this column always contains a zero.
- ZDEQ, which is the number of units of work taken from a zIIP queue by either the maintask or an MP subtask. This column has meaning only for the maintask and MP subtasks. For zIIP subtasks this column always contains a zero.
- ZIPCPU, which represents the total CPU time (in milliseconds) consumed by this zip engine. It is zero for MP subtasks and greater than zero when “z” precedes the task number and that zIIP subtask has been active (AMPSUBZ>0) at some point during the run.

Examples

Example 1 shows the results of the MONITOR STAT command. The zIIP statistics are **bolded and underlined** here for visibility.

Balancing channel load

AUDIT=16640 OUT=1225 IN=124 OUTXX=6836 INXX=182 DEV5=97 DEV6=581 DEV7=-
174 DEV8=30 WAIT=160665 DKRD=9078 DKWR=108152 SVRD=3766 SVWR=3775 CPU=44-
28 REQ=18 MOVE=117713 CNCT=556 SWT=19 RECADD=6362 RECDEL=10588 BADD=1443-
47 BDEL=288 BCHG=67379 IXADD=16116 IXDEL=31854 FINDS=661 RECDS=78622 DKA-
R=35853 DKPR=2579487 TFMX=387 USMX=13 SVMX=6 SFTRSTRT=1 DKPRF=864643 SMP-
LS=30 USRS=12933 SVAC=6000 BLKI=6000 PCPU=562 DIRRCD=29040 STCPU=3186 ST-
DEQ=23899 STWAIT=30743 STPOST=23952 LKWAIT=24 LKPOST=24 STIMERS=3831 SVPA-
GES=311303 COMMITS=3599 BACKOUTS=9 UBUFHWS=129448 TSMX=386 DKRDL=89 DKWRL-
=9611 CPUTOTZE=5 CPUONZIP=5 MTDEQ=116771 MTZDEQ=2082 STZDEQ=506 ZTDEQ=96
DKSRR=6792 DKSAAW=296 DKSRRFND=1921 DKSTBLF=3621 DKSTBLA=8 DKSTBLB=4874 -
DKSTBLC=3033 DKSTBLD=1725 DKSTBLX=48 DKSTBLE=13286

Example 2 shows output from a MONITOR TASKS command from the same run, using a Model 2 terminal:

```
12.121  APR 30  13.34.00          PAGE 2

TASK    CPU    PR    PCPU    WAIT    POST    LKWAIT    LKPOST    MOWTM    DEQ
  0      234    834    280     76     14         0         0      179     78
  1        0      6      0     10      8         0         0     8000      1
  2        0     10      0      1      1         0         0        0      0
z  3        0     16      0      2      1         0         0        0      0
z  4        0     19      0      1      1         0         0        0      0
```

Example 3 shows output from a MONITOR TASKS command from the same run, using a Model 5 terminal:

```
12.121  APR 30  13.33.35          PAGE 1

TASK    CPU    PR    PCPU    WAIT    POST    LKWAIT    LKPOST    MOWTM    DEQ    ZDEQ    SDEQ    LKWTIME    LKPTIME    ZIPCPU
  0      225    824    273     70     14         0         0     189     74     19     10         0         0         0
  1        0      6      0     10      8         0         0     8000      1         0         0         0         0         0
  2        0     10      0      1      1         0         0        0         0         0         0         0         0         0
z  3        0     16      0      2      1         0         0        0         0         0         0         0         0         0
z  4        0     19      0      1      1         0         0        0         0         0         0         0         0         0
```

Balancing channel load

You can balance channel loads for z/OS, z/VM, and z/VSE systems in the following ways:

- Balance I/O traffic on channels and control units by splitting the server data sets so that they occupy portions of more than one disk unit. Server data set allocation is discussed in “Allocation and job control” on page 171.
- Reduce memory requirements for Model 204 by using a single server for many users through the use of swapping (see “Requirements for server swapping” on page 169).

Note: Server swapping substantially increases the amount of disk I/O traffic and increases CPU utilization for the run.

- If main memory is not a problem and I/O or CPU is, you can provide each user with a server (NSERVS=NUMERS). The server data set and all associated I/O are eliminated.

- In a z/OS environment, IFAM2 and remote User Language performance is improved if the four CRAM SVC load modules are made resident. If all four modules cannot be made resident, performance is improved if the 1K module, IGC00xxx, is resident and the number of SVC transient areas increased.

For more information about the CRAM modules, see the *Rocket Model 204 z/OS Installation Guide*.

Reducing storage requirements

You can save a considerable amount of space by excluding unnecessary modules and relinking the various configurations of Model 204. Certain object modules are optional and can be excluded during link-editing and generation of load modules.

Reduce the size of Model 204 load modules by eliminating object modules supporting unused features. To facilitate identification of unnecessary modules, a list of object modules for each operating environment is included in the *Model 204 Installation Guides*.

Dynamic storage allocation tracking

The Dynamic Storage Allocation Tracking facility in Model 204 provides information about virtual storage allocation during Model 204 execution. When specifying region size and spare core space, take into consideration the statistics provided by the Dynamic Storage Allocation Tracking facility.

VIEW statistics

To display the following statistics during execution, issue the VIEW command. To display individual values, specify the value in the VIEW command, such as VIEW STORMAX, or specify the ALL or SYSTEM option, such as VIEW ALL.

Statistic	Meaning
STORCUR	STORCUR is the number of SPCORE storage bytes that are currently allocated. In z/OS, STORCUR includes storage above the line as well as below. It is possible for more storage to be freed than obtained post initialization, and a negative number in STORCUR is possible.
STORINIT	The amount of storage (in bytes) allocated during initialization. The STORINIT value does not include SPCORE space and the space for the Model 204 nucleus (ONLINE load module).
STORMAX	The high-water mark of STORCUR.

User interfaces

The user interface to dynamic storage tracking includes a message written to the audit trail and the CCAPRINT data set. To display parameter values during execution, issue the VIEW command.

The following message is displayed in the audit trail and the CCAPRINT data set:

```
M204.0090 DYNAMIC STORAGE ACQUIRED DURING  
INITIALIZATION=nnnnnn, AFTER INITIALIZATION=nnnnnn
```

Where

- The first value displayed is the amount of virtual storage acquired during system initialization. This value represents the minimum amount of storage needed for the parameters (except for SPCORE) specified in the CCAIN input.
- The second value is the high-water mark of storage acquired during operation after the initial allocation.

Additional storage can be acquired by some operating systems for control blocks and I/O areas (most notably z/VSE). Storage of this nature is not accounted for in Model 204 statistics.

Types of storage acquisition

Activities that can cause the operating system to get storage areas include:

- Using READ IMAGE from a VSAM data set (z/OS and z/VSE)
- Using sequential disk data sets through the USE command and User Language READ IMAGE and WRITE IMAGE statements
- CMS interface storage acquisition
- Using FLOD exits (discussed in the *Model 204 File Manager's Guide*)
- Using the Cross-Region Access Method (see Chapter 3, on IODEV=23 requirements)
- Using external security interfaces (see the *Rocket Model 204 Security Interfaces Manual*)

Cache fast write for CCATEMP and CCASERVR

The optional use of cache fast write is allowed for two heavily used data sets, CCATEMP and CCASERVR. The cache fast write feature, available on cached DASD controllers (such as IBM 3990 models 3 and 6) allows data to be written directly to a controller's cache. A write is considered successful as soon as it is accepted by the cache.

When the cache capacity is exceeded, parts of the data are downloaded to disk. Upon cache or DASD failure the disk image is not guaranteed; but since

temporary and server files are rebuilt during initialization, cache fast write provides increased performance with no integrity exposure.

To enable cache fast write, set the system initialization parameter:

CACHE=X'nn' :

where the valid values of *nn* are:

Value	Use cache fast write for...
'00'	Do not use; the default.
'01'	CCASERVER access
'02'	CCATEMP access
'03'	Both CCATEMP and CCASERVER access

Cache fast write options are selectable only at initialization and cannot be modified during the run.

Sequential processing

You can maximize sequential file processing performance by using file skewing or the Prefetch feature.

Prefetch feature

The Prefetch feature can improve performance of Model 204 for record-number order retrieval of a record set, particularly in a batch environment. Prefetch is for User Language applications only and applies only to Table B. It is not supported for Host Language Interface applications.

The Prefetch feature initiates a read of the next Table B page when a previous page is first declared to be current. The look-ahead reads are issued for the FOR EACH RECORD sequential record retrieval mode. Look-ahead read is suppressed if the FR statement contains an IN ORDER clause or if it reference a sorted set.

Performance gains resulting from the Prefetch feature vary, depending on the environment in which a run occurs.

For complete information on Prefetch, refer to the *Model 204 User Language Manual*.

How to use Prefetch

To use the Prefetch feature:

1. Resize the MAXBUF parameter, based on the following formula:

$$\text{MAXBUF} = \text{NUSERS} *$$

$(4 + 2 * (\text{Maximum FOR EACH RECORD loop nest level}))$

2. Turn on the Prefetch feature by setting the SEQOPT parameter to 1 on the User 0 line.

The Prefetch feature can also be controlled with the RESET command issued by a user having system manager privileges.

Long requests

You can trap and evaluate unexpectedly long requests from Online users before the resources required to honor the request are used. Set the parameters listed in Table 18-2 to define the maximum amounts of resources a request can use. Default settings, given in the *Model 204 Parameter and Command Reference*, are extremely large. A user can reset the parameters if the request is valid and exceeds the settings provided.

Table 18-2. Parameters for trapping long requests

Parameter	Setting	Meaning
MCNCT	300	Maximum clock time in seconds
MCPU	20000	Maximum CPU time in milliseconds
MDKRD	500	Maximum number of disk reads
MDKWR	250	Maximum number of disk writes
MOUT	1000	Maximum number of lines printed on the user's terminal
MUDD	1000	Maximum number of lines written to a directed output (USE) data set

If any maximum is exceeded, Model 204 pauses, displays the amount used, and issues messages that provide the option to continue or cancel the request. If the request is running under an application subsystem, it is canceled if any maximum is exceeded.

You can set the SCHDOPT parameter to X'10' to use the CSLICE parameter to verify that long request values are not exceeded.

IFAM2 control

Host Language Interface application programs in execution can dramatically affect system performance as seen by other IFAM programs or Online Model 204 User Language users. The amount of processing required for IFAM loops, hard-wait states, or noninteractive batch mode execution make it difficult to detect when the use of resources is excessive and service to other users degrades. Long request parameters are not available to Host Language Interface jobs.

Use the following commands to monitor and control the IFAM2 threads (IODEV=23 and IODEV=43). These commands are discussed in the *Model 204 Parameter and Command Reference*.

- IFAMCLOSE
- IFAMDRAIN
- IFAMFORCE
- IFAMHALT
- IFAMOPEN
- IFAMSTART
- IFAMSTAT

The following considerations apply:

- If possible, initialize every IFAM2 service program with support for at least one terminal. This allows use of the IFAM2 command facility.
- To avoid entering a line for infrequent monitoring purposes, specify User Language connections from host language programs to Model 204 in the IFAM/Model 204 service program for a teleprocessing monitor.
- If you issue one of the IFAM system control commands when IFAM2 is not active (no IODEV=43 statements in CMS, no IODEV=23 statements in z/OS or z/VSE), an error message is displayed.
- When using the IFAM control commands, if the CRAM argument is not specified and if any CRAM threads are defined, the CRAM channel is assumed.

If CRAM threads are not defined, the IUCV channel is assumed.

SQL system and user statistics

This section describes Model 204 statistics that are specific to Model 204 SQL processing. For information on SQL Server IODEV threads, see “SQL server threads (IODEV=19)” on page 94.

SQL statement statistics

Model 204 SQL processing generates a since-last statistics line after each Prepare, Execute, Open Cursor, terminate, and log-off.

The PROC field in the statistics output contains the SQL statement name. The LAST label indicates the type of statement:

- LAST = PREP indicates an SQL Prepare statement (compilation of an SQL statement), or compilation phase of Execute Immediate.
- LAST = EXEC indicates one of the following:

- SQL Execute (execution of a compiled statement)
- Execution phase of Execute Immediate
- Open Cursor (execution of the SELECT statements associated with a cursor) statement
- SQL user log-off

HEAP and PDL high-water marks

The HEAP statistic indicates the high-water mark for dynamic memory allocation (malloc) for processing of C routines. The SQL compiler/code generator uses this space. The SQL evaluator also uses the C heap for its runtime stack. Because dynamic memory is allocated before the system calls the SQL evaluator, the HEAP high-water mark does not change during a unit of SQL statement execution.

For user subtype '01' entries, the offset of the HEAP statistic is X '58', immediately following the OUTPB statistic. This statistic is also dumped by the TIME REQUEST command.

The PDL statistic, the pushdown list high-water mark, is checked and updated more often for SQL processing.

Interpreting RECDS and STRECDs statistics

The RECDS statistic (also used to count records processed by Model 204 FOR and SORT statements) indicates the number of physical cursor advances in an SQL base table and the number of records input to sort. The STRECDs statistic indicates the number of records input to SQL sort.

Multiple fetches of the same physical record without intervening cursor movement count as one read. For example, a sequential fetch of several nested table rows within a parent row increments the RECDS statistic by one.

Output sort records from SQL sort are not counted in the RECDS and STRECDs statistics.

Interpreting the PBRSFLT statistic

The Model 204 SQL sort uses a DKBM private buffer feature to increase the number of concurrent open buffers a user can hold. Use of the private buffer requires prior reservation. The PBRSFLT statistic indicates how many times a user failed to get a private buffer reservation.

In Model 204 SQL, reservation of this buffer is conditional, and the PBRSFLT counter is changed incrementally only for unconditional reservations. Therefore, PBRSFLT is always zero for current SQL users. Failure to obtain this buffer is not reflected in the statistic, but rather is indicated by a negative SQL code and message.

In the record layouts, PBRSFMT follows the SVPAGES statistic and is generated for system subtypes '00' and '01' and user subtypes '00' and '02'. For example, the offset of PBRSFMT for system type '00' entries is X '1E0'.

Interpreting the SQLI and SQLO statistics

The SQLI and SQLO since-last statistics record the high-water marks for the SQL logical input and output record lengths, respectively. They indicate the size of the largest SQL request to the Model 204 SQL Server and the size of the largest response from the SQL Server.

You can use SQLI and SQLO to size the buffers that receive and transfer Model 204 SQL requests and responses. These buffer sizes are set by the Model 204 CCAIN parameter SQLBUFSZ and the Model 204 DEFINE command parameters INBUFSIZE and DATALEN. These parameters are described in the *Model 204 Parameter and Command Reference*.

SQLBUFSZ must be greater than or equal to the maximum SQLI. The SQLI and SQLO high-water marks appear in since-last statistics and TIME REQUEST. They appear in the user since-last subtype x'01' statistics record at offset 92(x'5C') and 96(x'60'), respectively.

You can use both SQLI and SQLO to set INBUFSIZE and DATALEN to either minimize wasted buffer space or reduce traffic across the Model 204 SQL connection.

z/OS recovery

In z/OS, you can improve recovery performance by specifying the number of channel programs (NCP) and number of buffers (BUFNO) parameters in the DCB specifications for CCARF and RESTART streams.

- Multiple channel programs allow the initiation of multiple input requests at one time.
- Multiple input buffers allow input data to be processed with less I/O wait time.

Do not set BUFNO less than NCP:

- If BUFNO is less than NCP, Model 204 defaults BUFNO to the value of NCP.
- If BUFNO is set to a high value, SPCORE requirements are significantly increased.
- Rocket Software recommends that you set BUFNO to one more than the NCP value for more overlapped processing. For example:

```
//CCARF DD DSN=M204.JRNL1,DCB=(NCP=3,BUFNO=4),DISP=OLD
//RESTART DD
DSN=M204.CHP1,DCB=(NCP=7,BUFNO=8),DISP=OLD
```

Resident Request feature for precompiled procedures

QTBL (the Internal Statement Table for Model 204 procedures) and NTBL (which contains statement labels, list names, and variables) can occupy more than 40% of a user's server. Users executing shared precompiled procedures can reduce CCASERV and CCATEMP I/O by using shared versions of NTBL and QTBL in 31-bit virtual storage. If 31-bit storage is exhausted, Model 204 attempts to use storage below the line.

The system parameter RESSIZE controls the maximum amount of virtual storage allocated for resident requests. This storage is acquired dynamically on a procedure-by-procedure basis. The parameter RESTHRSH determines the number of server writes to be executed before a request is made resident.

STBL, NTBL, and QTBL reside at the end of the server. When a precompiled procedure incurs a number of server writes exceeding the value of RESTHRSH, NTBL and QTBL are copied into virtual storage rather than into the user's server space. Then another user executing the same procedure can use the copies of NTBL and QTBL that are already in storage, which considerably reduces server I/O every time a new user runs the request.

For general information on NTBL and QTBL, refer to "Labels, names, and variables table (NTBL)" on page 53 and "Internal statement table (QTBL)" on page 54 and to the *Model 204 User Language Manual*.

Performance considerations

The implementation of the Resident Request feature includes a mechanism (PGRLSE) for releasing unused storage associated with QTBL. This release storage strategy can have the effect of reducing Model 204 address space requirements for large Onlines, even if there is no server swapping.

Once a request is resident, it remains resident until the application subsystem is stopped. However, switching into nonresident mode (which disables the page release mechanism) can occur when:

- The next request in the program flow (determined by a global variable) is not resident.
- Users are swapped, and an inbound user is in nonresident mode.
- A request executes a Table B search or evaluates a FIND statement using a Numeric Range index. In this case, the request is loaded into the user's server to complete evaluation, but the resident copy remains.

The Resident Request feature is most effective when there is very little switching back and forth between resident and nonresident mode. Because frequent switching to nonresident mode decreases the overall effectiveness of the page release strategy, Model 204 automatically turns off the page release mechanism if the number of switches within a given subsystem exceeds 25% of the number of resident request executions. If the number of switches drops below 25%, then the page release mechanism is turned back on.

Storage protection for z/OS

If you want resident NTBLs and QTBLs to be storage-protected under z/OS, you must install M204XSVC. For more information, refer to

- “Labels, names, and variables table (NTBL)” on page 53
- “Internal statement table (QTBL)” on page 54
- “Activating IOS BRANCH ENTRY” on page 435.

System parameters

The following system parameters allow system managers to control and monitor resident requests:

- *RESSIZE* is a resettable User 0 parameter that defines the maximum amount of virtual storage (in bytes) to allocate for resident requests. The default is 0.
- *RESTHRSH* is a resettable User 0 parameter indicating the number of server writes that a request incurs before Model 204 saves its NTBL and QTBL in resident storage, assuming enough *RESSIZE* is available. The default is 100.
- *RESCURR* is a view-only system parameter indicating the amount of virtual storage (in bytes) that is currently being used to save resident NTBLs and QTBLs when running precompiled User Language requests.
- *RESHIGH* is a view-only system parameter indicating the maximum amount of virtual storage (in bytes) that has been used in this Model 204 run to save resident NTBLs and QTBLs when running precompiled User Language requests.

For complete information on these parameters, refer to the *Model 204 Parameter and Command Reference*.

SVPAGES statistic

To monitor the amount of data transferred as a result of server reads and writes, the SVPAGES statistic is available under the following journal entries:

- “User since-last statistics” on page 579
- “User final and partial statistics” on page 575
- “System final and partial statistics” on page 564

You can also view these statistics Online by using the MONITOR STAT and MONITOR SL commands.

Analyzing MONITOR command output

The following sample output shows how you can analyze resident request performance:

```
MONITOR SUBSYS (PROCCT) subsysname:

SUBSYSTEM:   PDS

NUMBER OF PRECOMPILABLE PROCEDURES (SAVED) :      15
NUMBER OF PRECOMPILABLE PROCEDURES (NOT SAVED) :   34
NUMBER OF NON-PRECOMPILABLE PROCEDURES :          14
NUMBER OF REQUESTS MADE RESIDENT :                 4
NUMBER OF ELIGIBLE REQUESTS NOT RESIDENT :          0
STORAGE USED FOR RESIDENT REQUESTS :               290816
NUMBER OF RESIDENT PROCEDURE EVALS :               110
NUMBER OF SWITCHES FROM RESIDENT MODE :            64
```

The “precompilable procedures not saved” have not been saved, because they have not been compiled or evaluated yet or, if they have, have not yet been involved in the number of server swaps specified by the threshold, RESTHRSH.

If “eligible requests not resident” is nonzero, this indicates that not enough RESSIZE was available to make them resident.

“Number of switches from resident mode” indicates the number of times the NEXT global variable invoked a request that was not resident, or the number of times the procedure being evaluated encountered a direct Table B search or a Numeric Range field and had to be loaded into the user’s server to complete evaluation.

Multiprocessing (MP/204)

MP/204, the Model 204 multiprocessing facility, makes full use of multi-processor configurations on IBM mainframes and compatibles running z/OS.

With MP, a single Online can access several processors simultaneously, allowing user requests to execute in parallel. Parallel processing increases throughput by giving the Model 204 address space additional CPU resources. An Online configured for MP can handle more volume in a fixed amount of time, or reduce response time for a fixed amount of work.

An MP Online performs parallel processing by distributing work between a maintask and multiple z/OS subtasks, which are attached during initialization. The subtasks (also called **offload tasks**) execute segments of evaluated code, including most User Language constructs. Certain activities can be performed only by the maintask; for example, external I/O including CHKPOINT and CCAJRNL.

The Model 204 scheduler and evaluator control the movement of work between the maintask and subtasks. The evaluator identifies constructs that can be offloaded, and requests task switching accordingly. Users ready for task switching are then placed on the appropriate queue. For example, when a subtask has finished executing a User Language FIND statement, the user is placed on a queue of users waiting to return to the maintask.

The amount of throughput gained by using the MP feature depends on several factors, including CPU resources available, the amount of work that can be offloaded, and the benefit of offloading compared to the cost of task switching. This section explains how to activate the MP feature, analyze performance statistics, and set MP tuning parameters.

CPU accounting

If your installation uses Timer PC (described in “Timer PC” on page 433), disable it before running an MP Online. This facility is incompatible with Model 204 MP implementation, which uses the standard z/OS task timer for CPU accounting.

Activating MP

Once MP/204 is installed, the actual use of offload subtasks is optional. To activate multiprocessing, set the following system parameters on User 0's parameter line:

- Setting the KOMMOPT parameter with the value X '01' creates multiple KOMMs, allowing multiple users to have private copies of a Model 204 data structure containing various arrays and pointers. This option reduces CPU usage, and is required for MP. Multiple KOMMs require approximately 4K additional storage per user.
- The NMPSUBS parameter determines the number of offload subtasks attached during initialization. This is normally set to a number less than the number of available processors. The maximum value allowed is 32.
For example, the normal setting of NMPSUBS for a three-processor hardware configuration is 2.
- The AMPSUBS parameter determines the number of active subtasks (subtasks that actually perform offloaded work). The AMPSUBS setting must be less than or equal to the value of NMPSUBS.

Unlike NMPSUBS, AMPSUBS can be reset by the system manager at any time by issuing the command RESET AMPSUBS n, where n is the desired number of active subtasks. This is useful for adjusting for fluctuating system usage, or for determining the optimum number of active subtasks by experiment.

It is also possible to turn off MP (in effect) by resetting AMPSUBS to the value 0. When AMPSUBS is 0, no work is offloaded (all work is performed by the maintask).

If you are using MP/204, consider “Using unformatted system dumps (z/OS)” on page 325.

In addition to the basic MP parameters, the SCHDOPT and MPOPTS parameters are useful for system tuning. For details, see “Scheduler operation and accounting (SCHDOPT)” on page 465 and “Optimization using MPOPTS and MP OPTIONS” on page 471.

The SCHDOFL parameter can also be used for system tuning, as needed, although the default value is recommended in most cases. For details, see “Scheduler tuning (SCHDOFL)” on page 466.

MP performance and tuning

MP statistics

The following list of system final and partial statistics measure various aspects of MP performance. STCPU and STDEQ are also maintained as user final, partial, and since-last statistics.

- CPU
- DKPR
- DKPRF
- LKPOST
- LKPTIME
- LKWAIT
- MPLKPREM
- MPLKWTIM
- MQWTM
- PCPU
- PR
- STCPU
- STDEQ
- STPOST
- STWAIT

These statistics are all written to the journal data set (CCAJRNL).

- For a description of each statistic see Table A-1 on page 548.
- For the journal record layout of MP statistics see Table A-14 on page 585.

Using MP statistics

- The CPU, PCPU, STCPU, and STDEQ statistics are important indicators of CPU utilization and task-switching overhead. These statistics are discussed separately in the next section.
- LKPOST and LKWAIT measure overhead associated with MP locks, that is, MP-specific mechanisms to regulate multiple users' access to facilities such as the disk buffer monitor.

You can change the number of spins that can be taken when MPLOCK is not available before going into wait by adjusting the setting of the MAXSPINS parameter.

Rocket Software recommends that you test the setting of the MAXSPINS parameter at your site. Increase the value of MAXSPINS, watching for a drop in the values of the LKWAIT and LKPOST statistics. When the values of the statistics stop dropping, set to that current MAXSPINS value.

- STWAIT and STPOST measure the number of WAIT and POST operations associated with MP subtask switching, and are, therefore, indicators of one component of MP overhead.

Fast logical page reads

A logical page read (DKPR) is a moderately expensive operation, especially in an MP/204 environment. Often a page is logically read repeatedly; each logical read incurs the same overhead as the previous logical read.

Logical page reads are optimized; a popular page is kept pending or in deferred release after use. If a page is open when a logical page read is done for it, the logical page read can be done very quickly. This type of logical page read is called a **fast read**, because of the shorter path length.

A fast read is tracked with the DKPRF statistic. A fast page read does not increment the DKPR statistic.

The relative ratio of fast reads to standard page reads might be improved by increasing the value of the MAXOBUF parameter and by setting the SCHDOPT X'04' bit as described in "SCHDOPT: MP/204 scheduler operation and accounting" and "MAXOBUF: Maximum number of open disk buffers per server" in the *Model 204 Parameter and Command Reference*.

Viewing MP statistics

In addition to analyzing the journal statistics described above, the system manager can view task-specific MP statistics directly by issuing the MONITOR TASKS command. For information on the MONITOR TASKS display, see the *Rocket Model 204 Parameter and Command Reference*.

Analyzing CPU utilization (CPU, STCPU, and STDEQ)

A logical first step in evaluating MP performance is to process a fixed amount of work with MP turned on and MP turned off, and then to compare results.

Normally the MP run consumes more total CPU than the non-MP run due to MP overhead, that is, extra CPU resources required for MP-specific activities such as task switching and MP resource locking.

If MP is working effectively, MP overhead is outweighed by the amount of processing that is performed in parallel by offload subtasks. Therefore, to analyze and tune MP, STCPU and related statistics must be considered in relation to the system and user CPU statistics.

Suppose, for example, that your comparison runs generate the following statistical data:

MP Turned Off:	MP Turned On:
CPU = 10000	CPU = 11000
	STCPU = 7000

These statistics show that your Model 204 application is a good candidate for multiprocessing. You can estimate total MP overhead by comparing CPU statistics: here, it is $11000 - 10000 = 1000$ milliseconds. As the STCPU statistic shows, 7000 milliseconds of Online processing is actually performed by offload subtasks; therefore, the actual CPU usage by the maintask is $11000 - 7000 = 4000$ milliseconds.

In the example above, the Model 204 application requires 10000 CPU milliseconds for processing (not counting MP overhead). The MP maintask does 4/10 of the total processing required. Therefore, MP throughput might improve by a factor of $10/4 = 2.5$.

Analyzing throughput

Use the following formula as an approximate measure of the **throughput potential** of a Model 204 MP application:

$$\text{NonMP CPU} / (\text{MP CPU} - \text{STCPU})$$

The throughput improvement actually achieved might differ from the throughput potential if any of the following constraints are present:

- Limited processor availability
If, for example, the throughput potential of an application is 2.5, at least three processors are needed to achieve it. If only one offload subtask is doing 60% of the work, then throughput is no better than it would be if the maintask were doing 60% of the work.
- CPU contention with other jobs

- Resource and locking bottlenecks in Model 204
- I/O bottlenecks (many I/O activities must be performed by the maintask)

STDEQ and STCPU

Another statistic useful for gauging subtask utilization is STDEQ, which counts the number of times that an MP subtask took a unit of work from the MP subtask queue and processed it. The ratio of STCPU / STDEQ is proportional to the number of offloaded instructions executed per time; so in tuning an MP application, it is a good sign if STCPU / STDEQ increases.

MP overhead

MP overhead (estimated by subtracting CPU without MP from CPU with MP) can be a performance concern for two reasons. First, overhead increases CPU demand for the processor configuration, which might be required to perform work other than Model 204 Online processing. Second, MP overhead can have a negative effect on response time.

The two main sources of MP overhead are MP locking and task switching:

- Locking overhead increases with CPU utilization. Most MP lock contention is due to buffer pool activity (measured by the DKPR, DKRD, and DKWR statistics, described in “Disk buffer monitor statistics and parameters” on page 435). Disk reads and writes (DKRD/DKWR) can be reduced by increasing the size of the buffer pool. The relative proportion of contention due to page requests (DKPR) vs. disk reads and writes depends upon the particular application and configuration.
- Task switching can be affected by a variety of strategies open to the system manager and applications programmer. These options are discussed in the next two sections.

Spinning on an MP lock

The MPSYS parameter keeps a thread spinning on an MP lock even if it appears that the task holding the lock cannot be dispatched. MPSYS=X'01' causes Model 204 to spin, which means loop on an attempt to get an MP lock such as the lock for the:

- Buffer pool
- Record locking table
- LRU queue, and so on

This loop repeatedly tries to acquire the lock by running for MAXSPINS times before issuing an operating system wait. Without MPSYS=1 Model 204 immediately issues an operating system wait if the lock is unavailable.

The overhead of issuing an operating system wait is significant. It is usually much higher than the overhead of spinning even 200 times. In most cases, Model 204 does not spin or loop 200 times before the lock becomes available. You will have saved significant overhead compared to issuing an operating system wait.

See “MPSYS: MP/204 processing options” in the *Model 204 Parameter and Command Reference* for more details.

Waiting on an MP lock

You can track MP lock waits with the MPLKPREM and MPLKWTIM statistics.

- **MPLKPREM:** Total elapsed time in milliseconds, across the maintask and all subtasks, the Online spent waiting due to operating system preemption.

This is the elapsed time between when an MP lock becomes available (lock post) making a task ready to run, and when the task actually gets the CPU. That preemption delay is caused by the operating system dispatching other tasks ahead of this task.
- **MPLKWTIM:** Total elapsed time in milliseconds, across the maintask and all subtasks, the Online spent waiting for MP locks.

Waiting on the operating system

LKPTIME: Lock preemption time—Total elapsed time in milliseconds, this task spent waiting, due to operating system preemption.

This is the elapsed time between when an MP lock becomes available (lock post) making the task ready to run, and when the task actually gets the CPU.

That preemption delay is caused by the operating system dispatching other tasks ahead of this task.

User Language considerations

When tuning an MP Online, it is useful to measure throughput potential for both frequently executed User Language requests and for the application as a whole. Programs can vary considerably in their ability to benefit from MP. Read-intensive applications can have very high throughput potential. Update-intensive applications tend to have lower potential, because most update operations cannot be offloaded. Also, specific rules govern offloading of structures such as loops, subroutines, and \$functions.

The MPOPTS parameter and the MP OPTIONS User Language statement allow the programmer or system manager to specify which program structures can be offloaded.

These options can have a direct bearing on task switching overhead, because excessive overhead is generally caused by offload subtasks executing too few

instructions to justify the cost of task switching. If certain offload strategies are not working efficiently, they can be disabled to reduce overhead.

Rules governing offloading, MPOPTS, and MP OPTIONS are discussed in the section “Optimization using MPOPTS and MP OPTIONS” on page 471.

Scheduler operation and accounting (SCHDOPT)

The Model 204 scheduler is responsible for task switching. The cost of each task switch tends to go up when subtask utilization goes down. When the amount of work in the subtasks is low, then z/OS WAIT and POST services are needed to complete the task switch. This can increase the cost per transaction; but this is usually an unimportant consideration when the system is lightly loaded.

The SCHDOPT parameter is available under z/OS and z/VM.

The default behavior of the Model 204 scheduler is to minimize overhead by switching users to offload subtasks only when other work is waiting for the maintask. This approach involves a trade-off between overhead and response time: with scheduler optimization, the maintask is often not available to process newly posted users as soon as possible.

If your processor complex has some excess CPU capacity, then you can improve response time by turning off scheduler optimization. Do this by setting the X '02' bit of the User 0 parameter SCHDOPT. With this setting, all possible work is offloaded, regardless of the current load on the maintask.

When scheduler optimization is turned off, it might be desirable to increase the values of the NMPSUBS and AMPSUBS parameters. If a trial run with SCHDOPT = '02' indicates that with n active subtasks, the maintask is consuming less than $1/n$ of the Online CPU, add 1 to the value of NMPSUBS and AMPSUBS.

Note: Optimization must be turned off (setting SCHDOPT = '02') in order to perform meaningful throughput potential analysis.

SCHDOPT settings X'00' to X'08' inclusive are used to control the maintask scheduler operation and account with MP/204.

Table 18-3. SCHDOPT settings affecting MP/204 processing

Setting	Purpose
X'00'	Changes the way since-last statistics are computed. X'00' is the default setting. This setting can make it appear that users are using more CPU time than is actually the case. If you want Model 204 to compute user since-last statistics without scheduler overhead, reset SCHDOPT to X'01'.
X'01'	Model 204 tracks the maintask scheduler overhead and generates the SCHDCPU statistic. This setting is used when the system manager does not want MP users to be charged for scheduler overhead.

Table 18-3. SCHDOPT settings affecting MP/204 processing (Continued)

Setting	Purpose
X'02'	Model 204 forces all processing that can be performed in parallel to run in an offload subtask, even if the maintask is not busy. The X'02' bit can be set whether server swapping is going to disk or to memory. It is most useful when swapping to a dataspace, because moving large amounts of data back and forth from a dataspace can be somewhat CPU intensive and justifies the cost of offloading the work to an MP subtask. However, keep in mind that setting the X'02' bit can introduce a slight delay into server swapping requests, although this is probably negligible.
X'04'	Model 204 defers closing disk pages on a critical file resource and other short-lived waits until the thread is actually swapped. Usually, when a swappable wait is done, Model 204 closes all still-open disk pages for that thread as quickly as possible. However, critical file resource waits tend to be short-lived, making this prompt page closing less efficient. SCHDOPT=4 means that deferred pages are released only at the time the user is swapped out. For all other SCHDOPT settings, deferred pages are released when the user is entering swappable wait state, even if server swap never takes place. If there is a chance that swappable wait does not take a lot of time, this option may be useful, because it saves several page reads. It is recommended that you experiment with this option.
X'08'	This setting is dependent on the setting of AMPSUBS; it has no effect if AMPSUBS=0. However, if you set AMPSUBS to greater than 0, then server swaps are performed in an MP subtask, if the maintask has any other work to do.

Because the difference between doing a fast read of a deferred release page and a regular logical page read is not that great, Rocket Software recommends that you adjust the SCHDOPT parameter X'04' setting only if you work in an MP/204 environment. Adjusting it does not have significant impact in other environments.

The SCHDOPT parameter is summarized in the *Model 204 Parameter and Command Reference*.

Scheduler tuning (SCHDOFL)

The SCHDOFL parameter indicates the target number of threads on the MP offload queue per active task. Setting SCHDOFL makes the Model 204 MP subtask scheduler more responsive to instantaneous increases in load.

SCHDOFL has a default value of 2, meaning that if there are more than twice as many units of MP offloadable work waiting to be processed in an MP subtask as there are active subtasks, and fewer than AMPSUBS subtasks are currently running, another MP offload subtask (not to exceed AMPSUBS) is activated.

Subtasks are deactivated, but remain available, when they have no work to do. This deactivation helps minimize MP overhead.

The default SCHDOFL value of 2 is recommended in most cases.

- Setting the value to 1 potentially increases throughput but also increases overhead, which is not recommended unless your site has plenty of spare CPU capacity.
- A value of 0 is not recommended because it will likely result in the unnecessary activation of MP subtasks -- by the time some of them are dispatched, other MP subtasks will have handled all pending units of work.
- A value greater than 2 might decrease CPU overhead while reducing throughput by reducing parallelism.

MP User Language statement processing

In an Online configured for MP, the User Language evaluator and Model 204 scheduler make decisions about what work to execute on the maintask and what work to offload to the subtasks.

The User Language programmer does not need to code special statements in order to make use of MP/204. However, different program designs do have different MP-related consequences. Therefore, measure and tune the MP performance of frequently executed requests at your Model 204 installation.

MP performance is optimal when the benefit of additional CPU utilization outweighs the cost of intertask communication. The evaluator contributes toward this performance goal by analyzing the statements and flow of control in User Language requests. In general, a programming construct is a good candidate for MP offloading if the number of instructions executed is considerably greater than the number of instructions required to perform task switching.

This section explains how the evaluator decides whether to offload specific User Language statements. Larger constructs are discussed in the next section.

Declarative statements

Several statements in User Language are not executable. These statements:

- Establish aspects of the environment (for example, VARIABLES ARE)
- Declare program constructs (for example, %variables and screen definitions)
- Define program structure (for example, SUBROUTINE and END statements)

These statements are ignored when the compiler makes decisions about the MP behavior of a request.

FIND statement

The FIND statement is always offloaded regardless of the context in which it is found. Two exceptions are:

- FIND AND PRINT COUNT is never offloaded.
- Table B searches are never offloaded. (The evaluator switches back to the maintask in order to perform this operation.)

When the evaluator encounters a FIND statement it switches from the maintask to a subtask (if not already in a subtask). If the FIND statement is in group context, then this switch is accomplished only once, prior to the FIND for the first file in the group.

Note: Table B searches in group context cause one task switch for each file that requires examination of Table B records.

If Table B searches are contributing to high maintask CPU consumption, then consider rewriting the request to perform the nonindexed selection criterion with IF statements in a FOR EACH RECORD loop (see “Tuning techniques: an example” on page 474).

SORT statement

Two phases are involved in processing the SORT statement: the extraction phase and the sort/merge phase. The extraction phase always takes place on the maintask. The sort/merge phase uses two different strategies depending on the number of records being sorted:

- For less than 20 records, the sort/merge takes place in the task context in effect when the SORT statement was encountered.
- For greater than or equal to 20 records, the sort/merge is always offloaded.

Mixed mode statements

Some statements can begin execution in a subtask, but might need to switch back to the maintask to perform certain operations (such as I/O). When the maintask work is complete, these statements switch back to their original task context. The following statements execute in this manner:

```
AUDIT  
BACKOUT  
CALL  
COMMIT, COMMIT RELEASE  
DELETE RECORD  
PAUSE  
STORE RECORD
```

Change in User Language READ SCREEN processing

MP/204 can process the {READ | REREAD | PRINT} {SCREEN | MENU} statements on an offload subtask, if all of the following conditions are true:

- The immediately previous terminal I/O was not in line mode (was not, for example, \$READ or PRINT), thus requiring that the end-of-page prompt be issued as part of READ SCREEN.
- No terminal I/O error, session loss, or SNA Communications Server (formerly VTAM) deactivation occurs during the execution of READ SCREEN.
- LAUDIT is not set to X'02' for LS audit lines.
- CAUDIT is not set to X'02' for CS audit lines.

Serial statements

The following statements execute exclusively on the maintask:

ADD	OPEN
CHANGE	OPENC
CONFIRM	POSITION
CONFIRMED	PRINT
CLOSE DATASET	PRINT ALL INFORMATION (except PAI INTO)
DELETE EACH	QUERY
DELETE	RECEIVE
DELETE RECORDS	RELEASE POSITION
FOR x VALUES	RESET HEADER
FILE RECORDS	RESET TRAILER
FIND AND PRINT COUNT	SEND
FIND VALUES	SET HEADER
FLUSH	SET TRAILER
FOR EACH VALUE	SIGNAL
INSERT	SKIP
NEW	TRANSFER

\$Functions

\$Functions operate in one of the following ways:

- MP=NO \$functions always execute on the maintask.
- MP=OK \$functions execute in whatever task context they are encountered.
- MP=YES \$functions always execute in an offload subtask.

Rocket Software does not deliver any MP=YES \$functions. The following Rocket Software-delivered \$functions are specified as MP=NO:

\$BLDPROC
\$CENQCT
\$DB2EMSG
\$ECBDGET
\$ECBDSET
\$ECBTEST
\$ENCRYPT
\$ENTER
\$FDEF
\$LSTFLD
\$LSTPROC
\$POST
\$READ
\$READINV
\$READLC
\$SLSTATS
\$SPIFY
\$STAT
\$UNPOST
\$WAIT

All other Rocket Software-delivered \$functions are specified as MP=OK.

MP program constructs

Main part of the request

If no serial statements are in the outermost nesting level, User Language requests begin execution in parallel. Otherwise, they begin execution on the maintask.

Loops

When the User Language compiler processes a FOR or REPEAT loop, it keeps track of the MP characteristics of the statements contained within the loop. It also keeps track of the MP characteristics of \$functions invoked within the loop. The compiler recognizes the following types of statements for these purposes:

- Serial statements
- Mixed mode statements
- All other statements

Serial and mixed mode statements are listed in the previous section.

If a loop contains any serial statements, then the evaluator is instructed to execute the entire loop on the maintask (with the usual exception for parts of FIND and SORT).

If a loop does not contain any mixed mode statements or MP=NO \$functions, then the evaluator is instructed to execute the entire loop on an offload subtask. Otherwise, the MPOPTS parameter or the MP OPTIONS statement (discussed below) determines how the loop is executed.

Loops that are coded with the use of JUMP statements and statement labels are not recognized as loops for the purpose of the compiler's MP analysis.

Subroutines

The compiler uses the same logic to keep track of the contents of subroutines as it uses for loops. The processing of CALL statements, however, is slightly more complicated due to the nature of the compiler.

The compiler's analysis of a CALL statement depends on whether the subroutine has already been compiled (a simple DECLARE for the subroutine has no effect). If the subroutine has not been compiled yet, then the compiler assumes that it contains serial statements and that the CALL statement requires a switch to the maintask. Otherwise, it uses the information actually collected about the subroutine. In other words, the CALL statement is treated as a mixed mode statement only if it precedes the subroutine definition, or if the subroutine actually requires serial execution.

The actual execution requirements of the subroutine are passed to the evaluator, just as for a loop. The CALL statement uses this information prior to invoking the subroutine. It can switch the user to the maintask, switch the user to an offload subtask, or perform no task switch at all.

ON UNITS

All ON UNITS receive control on the maintask. The loop and CALL statements contained within ON UNITS are handled with the same logic as loop and CALL statements outside of ON UNITS.

Optimization using MPOPTS and MP OPTIONS

When the User Language compiler analyzes the MP characteristics of a request, it must make decisions about how to handle the following situations:

- Nested loops
- MP=NO \$functions
- Mixed mode statements

The general question is whether or not the compiler "turns off" MP for a section of code in order to minimize task switching and the associated overhead. The compiler's default strategy is to move as much work as possible to offload

subtasks. The MPOPTS parameter and MP OPTIONS statement can be used by the programmer to modify this strategy in order to reduce the task switching overhead.

The MPOPTS parameter is a user resettable parameter whose default value is X '7F' (see Table 18-4 on page 472).

The MP OPTIONS statement is a User Language statement that enables an individual program to temporarily override the MPOPTS parameter.

MP OPTIONS statements can be inserted more than once in the same request. Each occurrence of MP OPTIONS controls the handling of all statements encountered until the next occurrence.

Syntax, parameters, and keywords are described in “MP OPTIONS/MPOPTS keywords and parameter settings” on page 472.

When to optimize

The use of MPOPTS/MP OPTIONS is normally unnecessary. Consider it only when:

- Overall MP overhead for the system is high (higher than 10-15%).
- STDEQ statistic is significantly higher than the DKRD statistic.
- Significant number of total system STDEQs can be traced to a single program or a small number of programs.

Coordinate the use of these options with the system manager responsible for performance measurement and tuning. See also the discussion on “Pros and cons of tuning User Language requests for MP” on page 475.

MP OPTIONS/MPOPTS keywords and parameter settings

The general syntax of the MP OPTIONS statement is:

MP OPTIONS ARE *keyword1 keyword2 ...*

where the keywords are described in Table 18-4.

Table 18-4. MPOPTS values and keywords for MP OPTIONS

MPOPTS/Keyword	Meaning
X '01'/LOOPS	LOOPS containing serial statements
X '02'/CALLS	CALLs to subroutines containing serial statements
X '04'/FUNCTIONS	MP=NO \$functions
X '08'/STATEMENT LEVEL UPDATES	STORE RECORD and DELETE RECORD statements

Table 18-4. MPOPTS values and keywords for MP OPTIONS

MPOPTS/Keyword	Meaning
X '20'/TERMINAL IO	READ, PRINT SCREEN/MENU, and REREAD statements
X '40'/EXTERNAL IO	READ and WRITE IMAGE statements
X '80'/TRACE	Produce compilation warning messages for serial statements

Loop

When a request has nested loops, the innermost loop might require maintask execution (it might contain a DELETE FIELD statement, for example). In this situation, the innermost FOR or REPEAT statement requires a task switch, if the outer loop does not contain any serial statements or MP=NO \$functions.

If the amount of work done for each iteration of the outer loop is small, then the decision to switch tasks each time the inner FOR/REPEAT statement is encountered is a poor one. In this case, the number of instructions executed by an offload subtask does not outweigh the cost of task switching. The programmer can override this decision by omitting the LOOPS keyword from the MP OPTIONS statement (or the equivalent bit from the MPOPTS parameter).

Calls

When a request calls a subroutine that requires serial execution, a task switch is required, if the CALL statement appears inside a loop or other subroutine running on the maintask. To override this default decision, omit the CALLS keyword from the MP OPTIONS statement (or the equivalent bit from the MPOPTS parameter).

Functions

A similar situation arises with regard to MP=NO \$functions. For example, if a user-written \$function requires MP=NO and this \$function is used frequently in expressions, then several task switches might be required for execution of a single statement. To instruct the compiler to treat MP=NO \$functions as serial statements, omit the FUNCTIONS keyword from the MP OPTIONS statement (or the equivalent bit from the MPOPTS parameter).

Mixed mode statements

Other mixed mode statements can also result in excessive task switching. The keywords for each type are:

Type	Keywords
Statement level updates	DELETE/STORE RECORD
Terminal I/O	READ/REREAD/PRINT SCREEN & MENU
External I/O	READ/WRITE IMAGE

There is currently no way of controlling the handling of AUDIT, BACKOUT, COMMIT, and PAUSE.

Special keywords

Two special keywords are on the MP OPTIONS statement. The CLEAR keyword turns off all other options, instructing the compiler to minimize task switching overhead. This option is mutually exclusive with the other options. The TRACE option tells the compiler to echo each statement in the request that requires execution on the maintask. This is useful for finding the cause of excessive task switching.

Tuning techniques: an example

Your knowledge of a program's structure, combined with information about the application data, can help you improve the MP performance of a User Language request. Consider the following code segment:

```
*****
Print all managers with compensation greater than
* $100,000 *
*****
FOR EACH RECORD IN EMP.MGR
  IF SALARY + BONUS GT 100000 THEN
    PAI
  END IF
END FOR
```

The PAI statement prevents the loop from executing in an offload subtask. This is a good decision, if most records satisfy the condition. However, if most records do not satisfy the condition, this is a poor decision.

You can change the evaluator's MP offload decision by enclosing the offending statement(s) in a REPEAT 1 TIMES loop, or by moving the statement(s) to a subroutine. Recode the previous example as follows:

```

FOR EACH RECORD IN EMP.MGR
  IF SALARY + BONUS GT 100000 THEN
*
* *****
*   * Note: repeat loop added for MP performance
*   *****
    REPEAT 1 TIMES
      PAI
    END REPEAT
  END IF
END FOR

```

The repeat loop causes the evaluator to execute the loop on an offload subtask and switch to the maintask only when the PAI statement is necessary.

Pros and cons of tuning User Language requests for MP

MP tuning techniques such as in the previous example have the following drawbacks:

- Program readability is compromised.
- Program requires additional QTBL space.
- Program requires additional PDL space.
- Execution of the program requires additional CPU.

Therefore, restrict this type of tuning to requests that account for a large percentage of the total application CPU requirement and to the sections of such requests that are most intensively used. In addition, clearly document any program structure whose sole purpose is to influence the MP behavior of the program.

Part VI

System-level Capabilities Available in Model 204

[Redacted]

[Redacted]

[Redacted]

This part describes the Model 204 system manager's role in working with files and procedures outside of Model 204.

19

Using HLI and Batch Configurations

In this chapter

- Overview
- IFAM 1
- IFAM2
- BATCH204
- BATCH2 utility

Overview

The configurations described in this chapter use the Model 204 Host Language Interface (HLI), which allows a user to invoke most of the system functions from application languages such as Assembler, COBOL, PL/I, and FORTRAN.

This chapter summarizes the Inverted Files Access Methods, IFAM1, IFAM2, and IFAM4, which are used as interfaces between application languages and Model 204; the BATCH204 configuration, which supports a single user without a terminal; and the BATCH2 utility, which establishes a remote connection with a Model 204 Online running in a separate region.

For additional information, see the following sections in other chapters:

- “Multiple jobs running on one CPU” on page 33

- “Setting user parameters” on page 64
- “File groups” on page 147
- “IFAM2 control” on page 452

For complete information about HLI, refer to the *Model 204 Host Language Interface Reference Manual* and the *Model 204 Host Language Interface Programming Guide*.

IFAM 1

IFAM1 is a batch configuration of Model 204 that supports Host Language Interface calls in Assembler, COBOL, FORTRAN, or PL/I to the Inverted File Access Method (IFAM).

Linking Model 204 modules with a user’s host language program is similar to linking with function subroutines. At runtime, the host language program is loaded dynamically with Model 204.

An IFAM1 application runs in its own region, which is always separate from the Online region. The program and Model 204 execute together in a single virtual machine or address space.

Because each IFAM1 application is linked to its own copy of the HLI/Model 204 routines, all IFAM1 programs must be relinked each time an HLI/Model 204 routine changes. By specifying the DYNAM compiler option, application developers can dynamically link the HLI/Model 204 portion of their application at runtime and obtain gains in efficiency of application maintenance and storage space.

For more information about IFAM1 jobs, see the *Rocket Model 204 Host Language Interface Reference Manual*.

z/OS and IFAM1

The following considerations apply to the z/OS JCL required to run IFAM1:

- Required JCL DD statements for IFAM1 are:
 - CCAPRINT
 - CCATEMP
 - STEPLIB
 - CCAGRP, if permanent file groups are used
 - CCASTAT, if security features are used
 - Host language program statements
- Optional DD statements are:
 - CCAAUDIT
 - CCAJLOG

- CCAJRNL
- CCASNAP
- SYSUDUMP

CCASNAP or SYSUDUMP is required if you want error diagnostics.

Note: CCAIN statements are not valid.

- EXEC statement parameters do not refer to Model 204 directly. Model 204 parameter settings and User 0 parameters are passed to Model 204 by the IFSTRT call, or by the IFSETUP call if using an IFDIAL thread.
 - PGM parameter (required) specifies the name of the user program that is being run, *not* IFAM1.
 - REGION parameter is optional. Calculate REGION size by determining the amount of memory that the application program requires and adding that to a minimum of 260K (buffers and other overhead). The Model 204 modules in IFAM1 occupy approximately 704K.
 - PARM field (used to set application runtime parameters) is optional. Model 204 parameters cannot be specified in the PARM field.
 - TIME parameter is optional.

z/VSE partition usage

An IFAM1 configuration uses the storage area between the end of the IFAM1 configuration and the beginning of the partition GETVIS area, unless the user-written host language program is PL/I or large enough for Model 204 to allocate some buffers from it. In the case of PL/I programs, some storage must be left between the end of the IFAM1 configuration and the beginning of the GETVIS area.

To enable Model 204 to take dynamic storage for the GETVIS area, set the SIZE parameter on the EXEC statement as follows:

SIZE=AUTO, nK

where *n* is the number of bytes to be left. The IFAM1 dynamic link load module that is loaded needs a large GETVIS area.

Figure 19-Figure 19-1. illustrates the partition storage division.

Figure 19-1. IFAM partition usage

PARTITION		
Low end		User-Written Host Language Program
		IFIF1DOS

Program area

SIZE parameter	IFAM1	Unused space reserved for dynamic storage allocation	GETVIS area
High end	Phase		

Running IFAM1 under z/VSE

The following considerations apply to running IFAM 1 under z/VSE:

- Do not specify Model 204 parameters on User 0's parameter line. These parameters are included in the IFSTRT call to HLI, or in IFSETUP.
- Specify a value for the SYSOPT parameter in this call in decimal format (for example, SYSOPT=16).
- Do not use the UPSI Job Control statement to specify bit settings for SYSOPT.
- You must catalog IFAM1 programs into a sublibrary before you can execute them.
- You must include label information for the CCATEMP, CCASTAT, CCAUDIT, and all user database files referenced by the IFAM1 program in the job stream:
 - EXEC statement refers to the phase name under which the IFAM1 program is cataloged.
 - Partition storage requirements must be met.

IFAM2

IFAM2 is a multi thread configuration of Model 204 that supports host language calls to HLI from one or more user programs. IFAM2 programs run as separate jobs in other regions or partitions and share a single copy of the database management system software. Each user program:

- Can have only one IFDIAL thread, although each program can have multiple IFSTRT threads.
- Must be linked to a small interface module in its region. The linked interface provides communication between the batch region (user program) and the IFAM region via a special inter-region supervisor call.

The IFAM2 feature can handle calls from several users at once, similar to the Model 204 ONLINE configuration. The ONLINE load module can be link-edited in such a way that it supports HLI user programs as well as terminals that run User Language requests (see the Rocket Model 204 Installation Guides).

Host Language applications can run in 31-bit addressing mode and move data to and from Model 204 in 31-bit addressing mode.

CRAM requirements for z/OS and z/VSE

The Cross-Region Access Method (CRAM) is required for z/OS and z/VSE systems.

If the host language program has been compiled with a z/VSE compile, or requires z/VSE SVC simulation, the z/VM/z/VSE environment (SET DOS ON) is also supported.

IUCV channels can be available in the same run under z/OS and z/VSE (prior to Release 1.4.0).

Length of HLI functions

The maximum length of an HLI function argument depends on the HLI application's language indicator:

- For PL/I, supply the length of each argument as part of the argument in a dope vector.
- For Assembler, COBOL, or FORTRAN, set the maximum length of an input argument using the LIBUFF parameter set on the user's parameter line.

The maximum length of an output parameter is the maximum value of LOBUFF.

IFAM buffer size (IFAMBS)

The value required for the IFAMBS (IFAM2 buffer size) parameter set on User 0's parameter line depends on the maximum string length and the highest number of string arguments used by a single HLI call.

IFAMBS is the maximum size of a block of data that can be transferred between the application and Model 204. The block contains a fixed header of 284 bytes, plus all the input and output arguments for a single function call. Four bytes are required for each integer argument. The maximum string length is required for each output string argument:

- For PL/I input strings, use the string's actual length.
- For COBOL and FORTRAN input strings, use the maximum length (LIBUFF). IFAMBS defaults to 2048 and can be set on User 0's parameter line.

To calculate IFAMBS, use the following formula:

$$\text{IFAMBS} = (n * \text{LIBUFF}) + 284$$

where n is the number of arguments used on a function call in your HLI program. The minimum value of n is 2.

Setting the IFAM2 channel name

Set the IFAM2 channel name on User 0's parameter line. For IFAM2 communication using:

- Cross-Region Access Method, set the IFAM2 channel name using the IFAMCHNL parameter.
- IUCV interface, set the channel name using the VMIFCHNL parameter.

Additional modules (IFAM, CRIO, IFII, IFIF)

To support IFDIAL threads, link-edit the CRIO module into the ONLINE configuration.

To generate an ONLINE load module that supports normal IFAM2 processing, CRIO processing, and provides z/OS support for CICS, link the following additional modules with ONLINE:

Module	Contains...
IFAM	Basic IFAM routines
IFII	Code to interface to Host Language programs
CRIO	Code to interface to User Language programs

If you need a special version of IFAM2, you can create the interface module IFIF without using the default channel names. Consult Technical Support for instructions.

Multiple IFAM2 versions

Multiple IFAM2 versions must use different names for each CRAM channel:

- If a second job is brought up without a unique name, you can initiate recovery with the following sequence of commands, if the second IFAM2 is initialized with User Language connections:

```

LOGIN system manager id
password
IFAMSTAT -1
IFAMDRAIN
IFAMSTAT
IFAMFORCE
IFAMCLOSE
IFAMOPEN newname
IFAMSTAT -1
IFAMSTART

```

- To terminate IFAM2 processing on a version of IFAM2 that has User Language connections, use the following sequence of commands, if no application jobs are active:

```
IFAMSTAT -1
IFAMDRAIN
IFAMSTAT
IFAMFORCE
IFAMCLOSE
LOGWHO
```

At this point, you can forcefully terminate active IFAM2 applications with the IFAMFORCE command, or allow normal completion by issuing the IFAMCLOSE command after you verify that the IFAM facility is drained (IFAMSTAT command).

Terminating IFAM2 processing

To terminate IFAM2 processing on a version of IFAM2 that does not have any IFDIAL connections, you must quiesce active IFAM2 applications (by canceling them or allowing them to complete).

If none of the applications are updating, or if broken files are acceptable, you can simply terminate IFAM.

Operational considerations when using the CICS and INTERCOMM system interfaces are described in the *Model 204 Host Language Interface Reference Manual*.

z/VSE and IFAM2

When using IFAM2 with ONLINE, Model 204 allocates data buffers in contiguous space. Model 204 attempts to allocate buffers between the end of the ONLINE program and the beginning of the partition GETVIS area first. If this storage area is not sufficient for a minimum amount of buffer space, it is left unused. You can keep the amount of storage in the unused area to a minimum by specifying SIZE=AUTO on the EXEC statement.

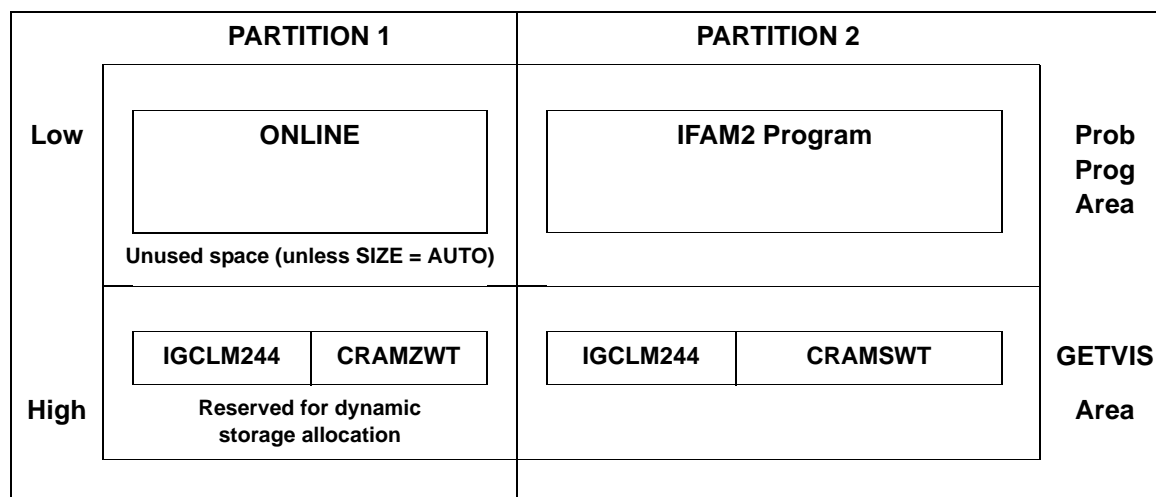
A situation might arise in which sufficient contiguous space is not found in either the reserved area or the area above it. In this case, specify either a very large or a very small SIZE VALUE. Figure 19-Figure 19-2. illustrates ONLINE partition usage with IFAM2.

In z/VSE, the partition in which IFAM2 application programs are run must have access to a core image library containing the phases IGCLM244 and CRAMSWT.

Label information is not needed for database files in the JCL for the IFAM2 application.

Label information for database files accessed by the IFAM2 application must be provided in the JCL of the ONLINE program that is communicating with the IFAM2 program.

Figure 19-2. ONLINE partition usage with IFAM2



z/VM and IFAM2

In z/VM, Model 204 files accessed by the host language programs are defined to the service machine:

- Define application program files in the user's z/VM virtual machine.
- You can use z/VSE macros within IFAM2 programs.
- You can run the copy of Model 204 communicating with the IFAM2 program under any operating system.
- IFAM2 permits user-written programs that use HLI and Model 204 to run in separate virtual machines. IUCV is the communications protocol.

IFAM4 for z/OS

IFAM4 is a multi thread configuration of Model 204 that supports host language (Assembler, PL/I, COBOL, FORTRAN) calls to HLI without the use of CRAM. Model 204 and the user.s application program are run as separate subtasks of a single batch job.

Note: IFAM4 is not applicable for z/VM or z/VSE. You must compile and link-edit IFAM4 before executing it. You must compile IFAM4 programs with the NODYNAM option.

Link-editing requirements

You must link IFAM4 with the REUS option:

- Name the resulting link IFAM4. (See the Rocket Model 204 installation guides for operating system instructions.)
- Link application programs into a load module library that is used by the IFAM4 run.

Examples

The COBOL compile and link procedure is:

```
// EXEC          COBUCL
.
.
.
//LKED.SYSLMOD DD DISP=(NEW,CATLG) ,
//              DSN=M204.PROGMS.IFAM4
//LKED.OB       DD DSN=M204.OBJECT,DISP=SHR
//LKED.SYSIN    DD *
                INCLUDE OB(IFIF4)
                NAME programname(R)
```

The PL/I compile and link procedure is:

```
// EXEC          PLILFCL
.
.
.
//LKED.SYSLMOD DD DISP=(NEW,CATLG) ,
//              DSN=M204.PROGMS.IFAM4
//LKED.OB       DD DSN=M204.OBJECT,DISP=SHR
//LKED.SYSIN    DD *
                INCLUDE OB (IFIF4)
                NAME programname(R)
```

Converting an IFAM2 application to an IFAM4 application

Only a link-edit, without recompiling the IFAM2 application, is required to convert an IFAM2 IFSTRT application program into an IFAM4 application. For example:

```
// EXEC          LKED
.
.
.
//LKED.SYSLMOD DD DISP=(NEW,CATLG) ,
//              DSN=M204.PROGMS.IFAM4
//LKED.OB       DD DSN=M204.OBJECT,DISP=SHR
//LKED.IFAM2LIB DD DSN=M204.PROGMS.IFAM2,DISP=SHR
//LKED.SYSIN    DD *
                INCLUDE OB (IFIF4)
                INCLUDE IFAM2LIB (oldprogram)
```

```
ENTRY entryname
NAME newprogram (R)
```

- Only the interface module (IFIF) for IFAM2 is changed.
- ENTRY statement is required.
- *entryname* must be the same as in the IFAM2 load module.

IFAM4 running JCL

IFAM4 requires essentially the same JCL as that described for ONLINE in “ONLINE data streams with CCAIN” on page 15.

- Use an IFAM4IN DD statement to specify a control data set.
- Your application program might require additional DD statements
- You must set the PGM parameter to IFAM4, not the name of the application program.
- REGION parameter must provide a region size large enough for both IFAM4 and the application program.
- PARM parameter must specify any parameters that are passed to Model 204.
- STEPLIB DD statement must specify the load module library or the library that contains the IFAM4 and the HLI application program. If you use separate libraries, concatenated DD statements are required.
- Set the NUSERS parameter on User 0.s parameter line to the maximum number of HLI threads needed by the application program plus one.
- Set NSERVS, if server swapping is required to save memory.
- An IODEV=23 statement, following User 0.s parameter line, is required for each HLI thread.
- Follow user parameter lines with a *SLEEP command that specifies a longer elapsed time (in seconds) than the application program needs to finish.
- Your application program might need SYSOUT or INFILE data sets. Avoid names that conflict with ddnames used for IFAM4 processing.
- IFAM4IN handles user application program parameters.

The IFAM4IN data set contains control statements used during IFAM4 initialization:

- The first statement (started in column one) contains the name of the application program as it was specified in the compile-and-link job.
- The second statement contains parameters that are passed to the application program. (These are the parameters normally specified in

the PARM field of the EXEC statement, if the application program is executed independently.)

- The second statement of IFAM4IN is omitted if the application program requires no parameters.
- The entire IFAM4IN data set is omitted if the application program is named APATTACH.

Common IFAM4 problems

Table 20-1 lists common IFAM4 problems and suggested solutions.

Table 19-1. Common IFAM4 problems

Problem	Solution
S806 ABEND	<ol style="list-style-type: none"> 1. Verify that IFAM4 has been named in the link-edit 2. Verify that the STEPLIB DD statement points to the libraries containing IFAM4 and the application program. 3. Verify that the application program name was correctly specified in column one of the first statement of IFAM4IN.
S804 or S80A ABEND	<ol style="list-style-type: none"> 1. Verify that REUS (RENT for VS1) is specified in the link-edit of IFAM4. 2. If REUS is not specified, relink IFAM4. 3. If REUS is specified, increase REGION by 10K and add 3000 to the SPCORE parameter. <p>SPCORE defaults to 12288 in IFAM4. Specify a setting (on the EXEC PARM field or on User 0.s parameter line) high enough to accommodate all the dynamic storage requirements of Model 204, the application program, and dynamically loaded subroutines used by the application.</p>

BATCH204

BATCH204 is a Model 204 configuration that supports a single user without the use of terminals (batch configuration). Input to BATCH204 is sequential, consisting of system control commands and User Language or FLOD statements. Output is a printed listing.

The BATCH204 interface is similar to the support offered to the HLI Interface in IFAM1 for IFSTRT. In the IFAM1 environment, the IFAM module is the interface between the application program and Model 204. In the BATCH204 environment, the IFID module is the interface between the application program and Model 204.

When using BATCH204, the JCL EXEC parameter is not available to Model 204. CCAIN and CCAPRINT files are used. The dynamic loading of Model 204 routines, such as the BATCH204 module, during initialization is the responsibility of the IFIF1OS, IFIF1DOS, and IFIF1CMS modules. One of these

operating-system-dependent modules must be linked with the application program.

BATCH2 utility

The BATCH2 utility provides the capacity to access a Model 204 Online running in a separate region. BATCH2 establishes a remote connection, emulates a line-at-a-time terminal, reads a specified input data stream, and transmits the input to the Online region. The resulting output generated by the Online region is read by BATCH2 and written to the specified output data stream.

In the z/OS and z/VSE environments, BATCH2 uses the CRAM thread defined by IODEV=29 (User Language line-at-a-time thread). In the z/VM environment, BATCH2 uses the IUCV line-at-a-time thread defined by IODEV=39.

A complete discussion, JCL requirements, and runtime parameters for the BATCH2 utility are given in the *Model 204 User Language Manual*.

Managing DCBLRECL to avoid an ABEND

If a BATCH2 job sends a line to CCAOUT that is longer than the DCBLRECL defined for CCAOUT, the line will be truncated at DCBLRECL-4 length to avoid an S002-18 abend. This only occurs for RECFM=V data sets. The following message will be issued for each line that exceeds the DCBLRECL length:

```
M204ULIF.0009: PREVIOUS CCAOUT LINE LONGER THAN DCBLRECL-4; LINE TRUNCATED
```

To avoid the error, either RESET OUTCCC to a value less than DCBLRECL-4 or increase DCBLRECL for CCAOUT.

20

Allocating and Directing Files Dynamically

In this chapter

- Overview
- Dynamic file allocation
- DEFINE DATASET command
- Extended Task Input/Output Table support
- ALLOCATE and FREE commands (z/OS and z/VM)
- Directed output
- USE command (z/OS, CMS, guest under z/VM)
- DEFINE PRINTER (z/OS, CMS, guest under z/VM)
- DEFINE PUNCH (z/OS, CMS, guest under z/VM)
- z/VSE output data sets
- USE PRINTER command (z/VSE under z/VM)
- USE PUNCH command (z/VSE)
- Using z/VSE/POWER
- Tailoring the Model 204 separator page

Overview

You can allocate new or existing files in the Model 204 system and direct the output to particular destinations without using specific data set definitions in the startup JCL.

This chapter explains the use of the following commands:

- DEFINE DATASET/PRINTER
- ALLOCATE
- FREE
- USE PRINTER/PUNCH

Examples for each operating system are provided.

After the command summaries are sections that explain how to direct output using z/VSE/POWER and how to print separator pages in block format.

For complete information about the Model 204 commands discussed in this chapter, refer to the *Model 204 Parameter and Command Reference*.

Dynamic file allocation

In z/OS and z/VM environments, you can perform dynamic file allocation in the following ways:

- Create a data set template with the DEFINE DATASET command in the CCAIN input stream before User 0's parameter line:

```
DEFINE DATASET name WITH SCOPE=SYSTEM  
other parameters
```

Then dynamically issuing the following command sequence:

```
ALLOCATE DATASET name WITH SCOPE=USER  
FREE DATASET name
```

- Allocate data set units in the JCL for the run and then dynamically issue the allocation command sequence:

```
ALLOCATE DATASET name WITH SCOPE=USER  
FREE DATASET name
```

In the z/OS environment, the actual data set allocation is performed by the standard SVC 99 routine provided by IBM. In the z/VM environment, the allocation service is performed by a Rocket Software-supplied SVC 99 emulation routine. For details on the SVC and the emulation SVC process, refer to the IBM *z/OS Job Management Manual*.

For z/VSE, files are allocated by complete data set definitions and ALLOCATE control statements in the JCL (see “ALLOCATE control statement” on page 611).

DEFINE DATASET command

The DEFINE command defines types of data structures, such as data sets, sequential I/O streams, program communication processes used by User Language requests, and output destinations.

The DEFINE DATASET command enables a user with system manager privileges to specify the physical characteristics of dynamically created data sets. DEFINE DATASET creates a template, identified by a name, which is referenced by ALLOCATE and FREE commands.

For a complete description of the DEFINE command and its variations, including DEFINE DATASET, refer to the *Model 204 Parameter and Command Reference*.

z/OS considerations

The following considerations apply when using the DEFINE DATASET command in a z/OS environment:

- The command can span more than one input line.
A continuation character (-) after the last parameter on the line is required.
- A newly issued DEFINE DATASET command replaces any previously defined duplicate names.
- SCOPE=SYSTEM is a required parameter indicating that the definition applies to all users on the system. SCOPE=SYSTEM can be issued only by a user with system manager privileges.
- The LABEL, POSITION, and DENSITY options of DEFINE DATASET apply only to dynamic allocation of tape data sets under z/OS. For examples, refer to the *Model 204 Parameter and Command Reference*.
- Operator messages and interactions for dynamic tape allocations are the same as for standard tape allocation. The interaction is controlled by the operating system.
- Parameters in a DEFINE DATASET command can be overridden by parameters specified in an ALLOCATE command. (See “ALLOCATE and FREE commands (z/OS and z/VM)” on page 498.)
- If you specify the STRNO parameter in the DD statement, you must give it the same value as the STRINGS parameter in the DEFINE DATASET statement. Preferably, do not specify STRNO in the DD statement.

Large data set support

Large data sets are supported for Model 204 database files and journal files, including CCAJLOG, GDG (z/OS only) and stream-based journals, and sequential data sets, including file dumps.

The Model 204 DEFINE DATASET and ALLOCATE DATASET commands include the LARGE argument that indicates a data set may have more than 64K tracks.

z/OS data sets may be SMS or non-SMS managed. A z/OS or z/VM data set must be sequential (defined with the PS option) to qualify as a large sequential data set.

Model 204 files may be dumped to or restored from large sequential data sets. Utilities for journal processing—UTILJ, MERGEJ, Audit204—support large format journal files as input data sets. MERGEJ also supports large data sets as output.

All options of the M204UTIL z/VM utility can allocate and erase large format data sets and initialize, write and read non-indexed OS-format VTOCs on large volume DASD. See the *Rocket Model 204 z/VM Installation Guide* for information on preparing and using OS format disks under VM.

EXCP and IOS Branch Entry support is also provided.

z/OS examples

The following examples illustrate the use of the DEFINE DATASET command.

- In this example, DEFINE DATASET creates a template. A user creates an output file that references the template with the ALLOCATE command:

```
DEFINE DATASET SAMPLE WITH SCOPE=SYSTEM -
LRECL=80 BLKSIZE=800 RECFM=FB SEQUENTIAL -
PRIMARY=1 CYLINDER VOLUME=RPG001
```

```
ALLOCATE OUTFILE LIKE SAMPLE WITH SCOPE=USER -
DSNAME=M204.VEHICLES.REPORT
```

- *LIKE* assigns the attributes of a previously defined data set or template to the data set being defined.
- *WITH* assigns the new attributes listed subsequent to it. WITH attributes override those specified in LIKE.
- In this example, the ALLOCATE command performs the allocation using the allocation attributes included in the DEFINE DATASET command. In this circumstance, ALLOCATE cannot specify any options.

```
DEFINE DATASET NAMEONE WITH SCOPE=SYSTEM -
LRECL=80 BLKSIZE=1600 RECFM=FB SEQUENTIAL -
PRIMARY=1 CYLINDER VOLUME=RPG000 -
```



```
DSNAME=M204.VEHICLES.REPORT
```

```
ALLOCATE NAMEONE (no attribute options)
```

z/VM considerations

The following considerations apply when using the DEFINE DATASET command in a z/VM environment:

- The command can span more than one input line.
A continuation character (-) after the last parameter on the line is required.
- A newly issued DEFINE DATASET command replaces any previously defined duplicate names.
- SCOPE=SYSTEM is a required parameter indicating that the definition applies to all users on the system. SCOPE=SYSTEM can be issued only by a user with system manager privileges.
- Parameters in a DEFINE DATASET command can be overridden by parameters specified in an ALLOCATE command. (See “ALLOCATE and FREE commands (z/OS and z/VM)” on page 498.)
- You can specify SECONDARY under z/VM, but no secondary allocation occurs when running under z/VM. The secondary space information is stored in the Volume Table of Contents (VTOC) when allocation is done under z/VM.
- The VOLUME option is required under z/VM when you create a new data set, or when you reference an existing data set that resides on an unaccessed-format or z/VSE-format minidisk. The VOLUME option is meaningful only during dynamic allocation.

Uses of DEFINE DATASET in the z/VSE environment

A DEFINE DATASET command in the z/VSE environment allows:

- Translation of the Model 204 internal 8-character file name into a 7-character z/VSE file name acceptable for use on a DLBL or TLBL statement
- Relating a Model 204 data set name to a logical unit number (useful when routing a data set to a file that cannot be referenced by a data set name)
- Routing output from a USE command to a printer, punch, or tape device (see “Directed output” on page 501)
- Specifying options (RECFM, LRECL) that cannot be coded in z/VSE JCL
- RESTART recovery of a Model 204 file in which the internal file name is related to a 7-character z/VSE file name

- Use of tape journals or checkpoint streams for RESTART recovery

z/VSE DEFINE DATASET options

The DEFINE DATASET options in a z/VSE environment are described in Table 20-1.

Table 20-1. DEFINE DATASET options in a z/VSE environment

Option	Specifies...
<i>name</i>	Internal name of either a sequential data set or a Model 204 file being defined.
DDNAME= <i>ddname</i> FILENAME= <i>filename</i>	Symbolic file name (for disk files) or symbolic unit number (for tape or unit record devices) used in the Job Control Language stream by which the data set can be referenced. The keywords DDNAME and FILENAME are synonymous.
RECFM= F FA FB FBA V VA VB VBA U UA	Record format of the data set.
LRECL= <i>logical-record-length</i>	Logical record length for the file. LRECL and BLKSIZE (below) are verified against each other when the data set is OPENed.
BLKSIZE= <i>physical-block-size</i>	Size in bytes of a block of data for the file.
DATALEN= <i>data-length</i>	True length of a data record. Record descriptor words for variable length records and/or carriage control characters are excluded.
DATALEN=0	Valid option for DEFINE DATASET. If the record length cannot be determined when the data set is opened, DATALEN=0 causes an error to occur.

z/VSE considerations

The following considerations apply to DEFINE DATASET in the z/VSE environment:

- The command can span more than one input line.
A continuation character (-) after the last parameter on the line is required.
- You can clear a data set definition by issuing another DEFINE DATASET command with the same name.
- Values you specify on the DEFINE DATASET command override default values provided by Model 204.
- Tapes used for sequential I/O must have a standard label and a TLBL JCL statement with a file name equal to the logical unit number. For example:

```
// JOB ONLINE
.
.
.
// TLBL SYS006,'tape file id'
// ASSGN SYS006,TAPE
.
.
// EXEC ONLINE,SIZE=AUTO
.
.
OPEN TEST
DUMP TO SYS006

or:

DEFINE DATASET DMPTEST WITH SCOPE=SYSTEM -
DDNAME=SYS006
OPEN TEST
DUMP TO DMPTEST
```

Extended Task Input/Output Table support

The z/OS operating systems Task Input/Output Table (TIOT) places a limit of 3,273 data sets (allocated dynamically or with DD statements) in any, not just Model 204, job step. To bypass this limit, you can use the Extended Task Input/Output Table (XTIOT) instead of the TIOT for certain data set types. There is no effective limit to the number of data sets that you can allocate using the XTIOT.

Using TIOT and XTIOT with data sets

- Every data set uses either TIOT or the XTIOT, never both.
- You decide whether to use the TIOT or the XTIOT on a per data set basis.
- The TIOT is always used for data sets:
 - Specified in the JCL through DD statements.
 - That are dynamically allocated as sequential or VSAM: that is, are not allocated as Model 204 files.
- In a RESTART recovery run, each data set that was dynamically allocated in the original run is reallocated, unless NODYNAM was specified. Reallocation always uses the same table, either TIOT or XTIOT, that was used for that data set in the original run.

Using TIOT and XTIOT with dynamically allocated Model 204 files

- Either the TIOT or the XTIOT can be used for dynamically allocated Model 204 files.

- Options XTIO and TIOT of the DEFINE and ALLOCATE commands determine which table to use.
- The SYSOPT2 parameter setting, X'80', determines whether to use TIOT or XTIO. See the *Rocket Model 204 Parameter and Command Reference* in the chapter on parameters P through Z for settings and behavior.
- You can specify the TIOT and XTIO only for Model 204 files with the OLD and DIRECT options on an ALLOCATE or DEFINE command, for example:

```
ALLOCATE DATASET PAYROLL WITH SCOPE=SYSTEM DSN=A.B.C.  
OLD DIRECT XTIO
```

- If TIOT is specified on the ALLOCATE command, the TIOT is used.
- If XTIO is specified on the ALLOCATE command, the XTIO is used.
- If neither TIOT nor XTIO is specified on the ALLOCATE command, SYSOPT2 determines whether the TIOT or XTIO is used. If SYSOPT2=X'80' is set, then the XTIO option is used, for only Model 204 file allocations, otherwise, the TIOT is used.
- Use of the XTIO requires use of IOS Branch Entry in the run, where XMEMOPT=X'02' is set in CCAIN. If IOS Branch Entry is not active and XTIO is specified, the following error is generated when the allocation is attempted and fails.

```
M204.2581 XMEMOPT=2 (IOS BRANCH) REQUIRED FOR XTIO  
OPTION
```

Special considerations for obsolete form of ALLOCATE

Special considerations apply when the following obsolete form of the ALLOCATE command is used:

Syntax `ALLOCATE DDNAME DSNAME`

Where:

- The TIOT and XTIO options cannot be specified.
- If the DDNAME starts with TAPE or OUT, sequential organization is assumed; the TIOT option is used.

Otherwise, a Model 204 file is assumed; either the TIOT or the XTIO is used, depending on the setting of SYSOPT2.

ALLOCATE and FREE commands (z/OS and z/VM)

The ALLOCATE command dynamically allocates new or existing Model 204 data sets or non-Model 204 data sets in z/OS or z/VM environments. See Appendix C for a discussion of the ALLOCATE utility in z/VSE.

To release control of data sets allocated either with ALLOCATE or in the startup JCL, use the FREE command.

Refer to the *Model 204 Parameter and Command Reference* for complete details of ALLOCATE and FREE syntax.

ALLOCATE command

When using the ALLOCATE command, the following considerations apply:

- For dynamic allocation units:
 - You must specify dynamic allocation units in the JCL for the run.
 - To calculate the maximum number of concurrently allocated units, add the value of the DYNAMBR parameter and the DYNAM statements in the run. ALLOCATE does not apply to VSAM.
 - ALLOCATE does not apply to VSAM.
 - When ALLOCATE completes, the file DCB area used is released.
 - The ALOCPRIV parameter (restricted to a user with system manager privileges) designates the data sets users can allocate. (See the *Rocket Model 204 Parameter and Command Reference*.)
The scope of the ALOCPRIV parameter is systemwide even when USER is the scope value.
- For primary and secondary space allocation:
 - Specify the allocation of space (BLOCKS, TRACKS, or CYLINDERS) for a new data set on the PRIMARY option. The BLOCK option allocates space by the BLKSIZE of the actual data multiplied by the PRIMARY option value.
 - Under z/OS, the operating system allocates a maximum of 15 secondary amounts of the type of space specified.
 - Secondary space allocation is not performed under z/VM, unless you specify the SECONDARY option and then you subsequently use the data set under z/OS.
- For Extended Address Volumes (EAV) allocation:
 - (z/OS version 1.12 and later) Extended Address Volumes (EAV) contain more than 64K cylinders. By specifying the EATTR parameter with the ALLOCATE or DEFINE DATASET command, you can allocate and use Model 204 database files and checkpoint, CCAJRNL, CCAJLOG, and USE data sets on EAV volumes above the 64K cylinder boundary. Both IOS Branch Entry (XMEMOPT=2) and EXCP I/O drivers can handle database files on EAV.
 - The EATTR parameter is effective only in z/OS version 1.12 and later; in all other environments it is ignored with no error message.

Note: Server data sets are not supported when allocated above the 64K

cylinder boundary. An attempt to open any server data set allocated above 64K cylinders results in issuing error message 2917, which identifies the server data set name, and run termination. See the *Rocket Model 204 Messages Manual* for details.

Command options

Except for SCOPE, the options of the ALLOCATE command are identical to the options of the DEFINE DATASET command. You can omit ALLOCATE options, if you have specified them on a previous DEFINE DATASET command.

Use of the SCOPE=SYSTEM option is restricted to a user having system manager or administrator privileges. Other users must specify USER or allow SCOPE to default to USER.

Data set types are indicated by file name prefixes:

Prefix	Data set type
OUT	USE data set (sequential organization)
TAPE	Deferred update data set (sequential organization)
DIRECT	Direct organization Model 204 file
CCA	Model 204 system data set (system manager privileges are required to allocate or free)

Options specified in the ALLOCATE command override DEFINE DATASET options.

Before performing an allocation, Model 204 verifies the NEW/OLD/COND, APPEND/NOAPPEND, and SHR/EXCL options.

The VOLUME option is required under z/VM when a new data set is created or when an existing data set that resides on an unaccessed z/OS-format or z/VSE-format minidisk is referenced.

Because no explicit file creation function exists in CMS, files on CMS disks do not require preallocation. However, you must preallocate files on variable-format disks with a primary allocation and optional secondary extents.

FREE command

The FREE command releases control of data sets allocated by the ALLOCATE command or by specifying the data set in the startup JCL. The freed data set disposition is controlled by the FREE/NOFREE and CATALOG/ NOCATALOG options specified on the ALLOCATE command.

A data set cannot be freed if one of the following conditions exists:

- Data set is currently in use.

- Data set is a Model 204 system data set (identified by a CCA prefix).

System manager privileges are required to use FREE unless the ALOCPRIV parameters allow the privilege to other users.

Directed output

You can direct output from Model 204 User Language requests and DISPLAY commands to destinations other than the normal Model 204 output device (usually the user's terminal). Alternate destinations are:

- Sequential data sets
- Dynamically allocated printer or punch device (z/OS or z/VM)
- Printer or punch device through the z/VM spooling system (guest operating system under z/VM)
- Printer using CICS as routing method
- Printer or punch device using z/VSE/POWER as a router
- Dynamically allocated printer, punch, or internal reader (z/OS, z/VM, or a guest operating under z/VM)

Note: This functionality is obsolete and supplied only for compatibility.

Command Overview

The following commands are required to direct output:

- DEFINE PRINTER or DEFINE PUNCH to establish definitions (a template) for alternate output destinations

DEFINE PRINTER/PUNCH commands can be issued only by a user with system manager privileges and can be overridden by another DEFINE PRINTER/PUNCH command for the same template name.

- USE PRINTER or USE PUNCH to direct output to a dynamically allocated printer or punch device

Any user can issue the USE command. Parameters specified on a DEFINE PRINTER/PUNCH command can be overridden by the USE command.

If the output is directed to a sequential data set, a DD/ DLBL/ TLBL/ FILEDEF statement for this data set must be in the Model 204 JCL associated with the user. (Under z/OS or CMS, the sequential data set can also be dynamically allocated during the run using the ALLOCATE command.)

For syntax and examples of the DEFINE and USE commands, refer to the *Model 204 Parameter and Command Reference*.

USE command (z/OS, CMS, guest under z/VM)

When you issue the USE command to direct output dynamically to a printer or punch under z/OS, CMS, or a guest operating system under z/VM, you can specify output options by the WITH keyword. Options specified by WITH override corresponding options specified in the DEFINE command.

You must specify the ROUTER option as z/OS, z/VM, or CICS in the USE command or in the DEFINE (PRINTER or PUNCH) command referenced by the USE command.

Parameters valid with the ROUTER=CICS option are:

- HDR1
- HDR2
- SCOPE
- SEP/NOSEP
- TERMID (terminal identification of the printer)
- TRANSID (transaction name of the printer transaction)

DEFINE PRINTER (z/OS, CMS, guest under z/VM)

The DEFINE PRINTER command creates a template that specifies the characteristics of printer output. The template is used for the life of the job unless it is overridden by another DEFINE PRINTER command for the same template name.

Use DEFINE PRINTER in conjunction with the USE command to direct output to a specified printer.

z/OS considerations

Under z/OS, directed output requires a JCL DD statement that associates a CCAPR data set with an alternate device (usually a printer). For example:

```
//CCAPR DD UNIT=device unit number
```

The device must be included in both the SYSGEN and the directory for the virtual machine is in use.

The following example relates each CCAPR to a virtual print device:

```
//CCAPR DD UNIT=015  
//CCAPR1 DD UNIT=016  
.  
.  
.  
//CCAPR7 DD UNIT=022
```


When you use the DEFINE PRINTER command to establish definitions for alternative output destinations, increase the amount of spare core to allow for the definitions.

CMS considerations

Under CMS, the M204SPL EXEC establishes the link to the z/VM spool for printed output. The EXEC can support all the options for the DEFINE PRINTER and USE PRINTER commands. For more information on the M204SPL, see the *Rocket Model 204 z/VM Installation Guide*.

DEFINE PUNCH (z/OS, CMS, guest under z/VM)

The DEFINE PUNCH command creates a template that specifies the characteristics of punch directed output. The template persists for the life of the job unless overridden by another DEFINE PUNCH command for the same template name.

Use DEFINE PUNCH in conjunction with the USE command to direct punch output.

The following considerations apply:

- Punch output requires a JCL DD statement that associates a CCAPR data set with the address of a punch device.
- A device is allocated to a user for the duration of the USE command.
- You must associate each device name with a unique unit address by specifying CCAPU data sets (CCAPU through CCAPU99) in the JCL. You can use all the specified punch data sets concurrently.
- You must include the device address in both the z/OS SYSGEN and, if running as a guest under z/VM, the z/VM directory for the VS1/SYS/z/OS virtual machine. Each CCAPU data set must be related to a virtual punch.

The following example associates each device name with a unique unit address.

```
//CCAPU DD UNIT=045
//CCAPU1 DD UNIT=046
.
.
.
//CCAPU99 DD UNIT=145
```

z/VSE output data sets

The following requirements pertain to z/VSE output data sets:

- Any data set routed to disk must be described with DLBL and EXTENT statements in the JCL stream executing the batch or online program.

- Any data set routed to tape must be described with a TLBL statement in the JCL stream used to execute the batch or online program.
- If a data set is directed to a printer or punch device, a DEFINE DATASET command must be issued prior to the USE command to set the logical record length, the block size, and the record format.
- The record format must be F (fixed unblocked).

Example

In this example, the USE command directs output of one procedure to a disk file, and the output of a second procedure to a printer. The keyword PRINTER is omitted in the USE command for the second procedure:

```
// JOB ONLINE WITH USE COMMAND
// DLBL OUTPRT,'USE.OUTPRT.FILE'
// EXTENT SYS002,balance of extent information
// ASSGN SYS002,X'cuu'
// DLBL TEST,'M204.DATA.BASE.',,DA
// EXTENT SYS005
// ASSGN SYS005,X'cuu'
// ASSGN SYS010,PRINTER
// UPSI 10010000
// EXEC ONLINE,SIZE=AUTO
PAGESZ=6184
LOGIN system manager id
password
OPEN TEST
USE OUTPRT          <---- Directed output to a disk file
INCLUDE ACCOUNT.REPORT
.
.
DEFINE DATASET OUTFILE WITH SCOPE=SYSTEM FILENAME=SYS010
USE OUTFILE          <---- Directed output to printer
INCLUDE ACCOUNT.REPORT2
/*
/&
```

USE PRINTER command (z/VSE under z/VM)

The following requirements apply to z/VSE running under z/VM:

- Define print data sets (CCAPR) using the DEFINE DATASET command.
- Associate the CCAPR file and the print device through a JCL ASSGN statement.
- If a data set is directed to a printer or punch device, issue a DEFINE DATASET command prior to the USE command to indicate the logical record length, record format, and block size.

- Define the print or punch device in the z/VSE supervisor and, under z/VM, the directory for the virtual machine.

Example using the JCL ASSGN statement

This example shows the association of the CCAPR data set with the device through a JCL ASSGN statement and a DEFINE DATASET command. The keyword PRINTER is omitted in the USE command:

```
// JOB ONLINE
// ASSGN SYS021,X'06E' ---- Directed output printer
// ASSGN SYS022,X'07E' ---- Directed output printer
.
.
// EXEC ONLINE, SIZE=AUTO
LOGIN system manager id
password
DEFINE DATASET CCAPR with SCOPE=SYSTEM FILENAME=SYS021
DEFINE DATASET CCAPR1 with SCOPE=SYSTEM FILENAME=SYS022
DEFINE PRINTER THIRDP with SCOPE=SYSTEM -
ID=RSCS,TAG=THIRDP,ROUTER=VM
.
.
USE THIRDP
.
.
.
/*
/&
```

USE PUNCH command (z/VSE)

The following requirements apply to USE PUNCH in a z/VSE environment:

- Define punch data sets (CCAPU through CCAPU7) using the DEFINE DATASET command.
- Associate the CCAPU file and the punch device through a JCL ASSGN statement.

You must define the punch device, X 'cuu', in both the z/VSE Supervisor and the z/VM directory for the z/VSE virtual machine.

Example 1: USE PUNCH in z/VSE

This example shows how to define punch-directed output in a z/VSE environment:

```
// ASSGN SYSxxx,X' cuu'
.
.
```

```
.  
// EXEC ONLINE,SIZE=AUTO  
PAGESZ=6184,NUSERS=  
.  
.  
.  
LOGIN system manager id  
password  
DEFINE DATASET CCAPU WITH SCOPE=SYSTEM FILENAME=SYSxxx
```

Example 2: USE PUNCH in z/VSE running as guest under z/VM

This example illustrates how to define punch-directed output for z/VSE running as a guest under z/VM. The keyword PUNCH is omitted in the USE command:

```
//JOB ONLINE  
.  
.  
.  
//ASSGN SYS021,X'06E' DIRECTED OUTPUT PUNCH  
//ASSGN SYS022,X'07E' DIRECTED OUTPUT PUNCH  
.  
.  
.  
//EXEC ONLINE, SIZE=AUTO  
//CCAPU1 DD UNIT=046  
.  
.  
.  
LOGIN system manager id  
password  
DEFINE DATASET CCAPU WITH SCOPE=SYSTEM FILENAME=SYS021  
DEFINE DATASET CCAPU1 WITH SCOPE=SYSTEM FILENAME=SYS022  
.  
.  
.  
DEFINE PUNCH THIRDP WITH SCOPE=SYSTEM -  
ID=RSCS,TAG=THIRDP,ROUTER-VM  
.  
.  
.  
USE THIRDP  
.  
.  
.  
/*  
/&
```

Using z/VSE/POWER

The implementation of directed output under z/VSE/POWER requires a DEFINE DATASET CCAPPRn/CCAPPU_n command to specify a logical unit name that is used in a JCL ASSGN statement to define a POWER printer or punch device. Printers and punches used by z/VSE/POWER must be defined during IPL and started for each partition using the following specifications:

```
PRINTERS=cuu1 through cuu8
```

and

```
PUNCHES=cuu1 through cuu8
```

A device is allocated to a user for the duration of the USE command.

Printer and punch devices

Simultaneous directed output is supported by the use of CCAPPR and CCAPPR1 through CCAPPR7 for printers; CCAPPU and CCAPPU1 through CCAPPU7 for punches. Each CCAPPR and CCAPPU must be associated with a unique logical unit. The number of defined devices determines the number of simultaneous USE PRINT or USE PUNCH commands that can be directing output.

To identify printer and punch devices for z/VSE/POWER:

```
DEFINE DATASET powername WITH SCOPE=SYSTEM, -  
DDNAME=SYSxxx
```

where *powername* for printers is:

```
CCAPPR, CCAPPR1, . . .
```

and where *powername* for punches is:

```
CCAPPU, CCAPPU1, . . .
```

z/VSE/POWER JCL example

This example shows a z/VSE/POWER JCL:

```
* $$ JOB JNM=MODEL204, CLASS=2  
// JOB MODEL204 ONLINE (EXECUTES IN PARTITION F2)  
.  
.  
.  
// ASSGN SYS011,X'02E'          <---- POWER printer 02E  
// ASSGN SYS012,X'03E'          <---- POWER printer 03E  
// ASSGN SYS013,X'04E'          <---- POWER printer 04E  
// ASSGN SYS021,X'01D'          <---- POWER punch
```

```
// ASSGN SYS022,X'02D'          <----- POWER punch
// EXEC ONLINE,SIZE=AUTO
DEFINE DATASET CCAPPR WITH SCOPE=SYSTEM,DDNAME=SYS011
DEFINE DATASET CCAPPR1 WITH SCOPE=SYSTEM,DDNAME=SYS012
DEFINE DATASET CCAPPR2 WITH SCOPE=SYSTEM,DDNAME=SYS013
DEFINE DATASET CCAPPU WITH SCOPE=SYSTEM,DDNAME=SYS021
DEFINE DATASET CCAPPU1 WITH SCOPE=SYSTEM,DDNAME=SYS022
LOGIN system manager id
password
DEFINE PRINTER PRT1 WITH SCOPE=SYSTEM,ROUTER=POWER,      -
CLASS=A,NOHOLD,HDR1=OUTFILE1
DEFINE PRINTER REMOTE WITH      SCOPE=SYS-
TEM,ROUTER=POWER,CLASS=Y
DEFINE PRINTER PRTX LIKE PRT1 WITH FORMS=3PRT
DEFINE PUNCH CARD WITH SCOPE=SYSTEM,ROUTER=POWER,HOLD    -
DEFINE PUNCH INTERNAL WITH SCOPE=SYS-
TEM,ROUTER=POWER,INTRDR,CLASS=G
DEFINE PUNCH REMCARD LIKE CARD WITH ID=015
USE PRT1 WITH NAME=MYOUTPUT
.
.
USE MYJOB LIKE INTERNAL WITH NAME=MYJOB
.
.
/*
/&
* $$ EOJ
```

z/VSE/POWER system requirements: the \$\$BSGMNT transient

The IBM-supplied \$\$BSGMNT transient must be available to Model 204 to perform directed output using z/VSE/POWER.

Information for printer and punch directed output (* \$\$LST statement and * \$\$PUN statement) is passed to the \$\$BSGMNT transient. Refer to the IBM *z/VSE/POWER Installation and Operations Guide* for a detailed description of the * \$\$LST and * \$\$PUN statements.

Options specifying default values are not passed to the \$\$BSGMNT transient.

z/VSE/POWER job stream

When using z/VSE/POWER, take note of the following considerations:

- The DEFINE DATASET command can appear before User 0 input parameters.

- DEFINE PRINTER and DEFINE PUNCH must appear after User 0 input parameters and are restricted to users with system manager or administrator privileges.
- If 'G' is a valid execution class for one of the z/VSE/POWER partitions, the punched output from 'USE PUNCH MYJOB' is run by z/VSE whenever a class G partition is free.
- The total length of the power statements cannot exceed 71 characters. This includes the 9 characters required for '* \$\$ LST' or '* \$\$ PUN'. Parameters and values on the DEFINE PRINTER/PUNCH or USE PRINTER/PUNCH commands, after translation to conform to the power statements, cannot exceed 62 characters.
- If you define an option with a null value on a DEFINE PRINTER/PUNCH or USE PRINTER/PUNCH command, the parameter is not used. This is useful when you specify the LIKE option of these commands.
- The keyword WITH denotes that specific directed output parameters follow.
- Parameters can be separated by commas or blanks.
- The LIKE keyword copies the attributes of some other template previously defined in a DEFINE PRINTER command.

LIKE used by itself copies all the attributes of the previously defined template specified. WITH used in combination with LIKE causes the parameters specified after WITH to override the attributes of the template name specified in the LIKE phrase.

Tailoring the Model 204 separator page

Use the option to print separator pages in block character format in conjunction with the SEP and HDR1, HDR2, and HDR3 parameters.

To use this option, the IBM-supplied routine, IEFSD095, must reside in the system library SYS1.AOSB0 and must be linked in the ONLINE load module. Refer to the *IBM Systems Modifications Manual* for more information about the routine.

21

Accessing BSAM and VSAM Files

In this chapter

- Overview
- Sequential I/O processing
- VSAM I/O processing

Overview

The sequential and VSAM I/O processing facility allows User Language requests to access BSAM and VSAM files through an image format defined in User Language requests. Access to BSAM files is supported in all versions of Model 204. Access to VSAM files is supported in z/OS and z/VSE versions of Model 204.

Images allow reads and processing of sequential files, VSAM files, and terminal input (IFDIAL connections), writes to sequential files and terminals, and creation of formatted blocks of storage used to build print records or process data that is in a non-Model 204 format. (For more information about images, see the *Rocket Model 204 User Language Manual*.)

This chapter discusses how Model 204 accesses basic sequential and VSAM files and the commands and system specifications that the I/O processing facility requires.

Sequential I/O processing

The sequential I/O facility has an external file I/O support feature that allows READ access to any sequential file whose organizational structure is in accordance with the BSAM data set structure (z/OS) or the SAM file structure (z/VSE):

- In z/OS operating systems, Model 204 can obtain BLKSIZE, LRECL, RECFM, BUFNO, and NCP information from the DCB parameter of the DD statement or from the disk file label at the time the data set is opened. A DEFINE DATASET command is not required.
- In z/VSE operating systems, you must issue a DEFINE DATASET command for each sequential file to be accessed and you must include the BLKSIZE, LRECL, and RECFM parameters.
- In z/VM, only one user at a time can access a particular sequential file. A CLOSE DATASET command must be issued between user opens.

If a previous user has not closed the file, any subsequent attempt to open the file brings down the z/VM virtual machine.

Buffer requirements

Sequential I/O processing requires a number of buffers equal to the BUFNO parameter of the DEFINE DATASET command for each concurrent OPEN DATASET command issued.

The buffer size required for a file is equal to the block size of the file. Buffer space is released when the file is closed.

Determining sequential data set format

Model 204 determines the format of sequential data sets by determining the values of the record format, data length, logical record length, and block size of the data set when it is opened.

- The **record format** describes the way physical records are structured. Attributes recognized by Model 204 are:

Attribute	Meaning
F	Fixed length
V	Variable length
U	Undefined length
B	Records are blocked
A	Carriage control present

The allowed value combinations are:

F, FA, FB, FBA, V, VA, VBA, U, UA

- **Data length** is the maximum amount of data that can be placed into a logical record.

Data length does not take into account carriage control characters, record descriptor words, or block descriptor words.

- **Logical record length** is the maximum length (in bytes) of each record in the data set.

The record descriptor word and carriage control character are included if they are present. If the records are of variable or undefined length, the maximum logical record length must be specified.

- **Block size** is the maximum length (in bytes) of a block.

If the record format is F, the block size must be an integral multiple of the logical record length. If the record format is V, the specified block size must be the maximum block size and at least four bytes larger than the logical record length.

Sequential data set algorithm

Model 204 determines the sequential data set characteristics by using an algorithm based upon values derived from:

- Operating-system-dependent information
- Data set definition (created by the DEFINE DATASET command)
- Information provided by Model 204 that is dependent upon the data set being opened

The steps of the algorithm are performed in order, with each successive value dependent upon the preceding values. Undetermined values are systematically defaulted. Erroneous values are flagged. Values (such as logical record length) are changed in order to make them consistent with previously examined values (such as record format and data length).

Table 21-1 summarizes the characteristics examined by the algorithm and the action taken.

Table 21-1. Algorithm for determining sequential data set characteristics

If the characteristic examined is...	And the value found is...	Then the result is...
Record format (RECFM)	No value	Undefined record
	RECFM =	Undefined record
Data length (DATALEN)	DATALEN = 0, RECFM = U	DATALEN = BLKSIZE

Table 21-1. Algorithm for determining sequential data set characteristics (Continued)

If the characteristic examined is...	And the value found is...	Then the result is...
	DATALEN=0, RECFM=U, carriage control	DATALEN = DATALEN - 1
	Negative DATALEN, RECFM = U	OPEN fails
	DATALEN = 0, LRECL not 0	DATALEN = BLKSIZE
	DATALEN = 0, LRECL not 0, zero, carriage control	DATALEN = DATALEN - 1
	DATALEN = 0, LRECL not 0, RECFM = V	DATALEN = DATALEN - 4 (if blocked) DATALEN = DATALEN - 8 (if unblocked)
	Negative DATALEN, LRECL not 0	OPEN fails
	DATALEN = 0	OPEN fails (no default)
Logical record length (LRECL)	LRECL < 32767	LRECL = DATALEN
	LRECL < 32767, carriage control	LRECL = LRECL + 4
	LRECL > 32767	OPEN fails
Block size (BLKSIZE)	BLKSIZE > LRECL	BLKSIZE = LRECL
	RECFM = F	BLKSIZE = multiple of LRECL (rounded down)
	RECFM = V	BLKSIZE = LRECL + 4
	BLKSIZE > 32767	OPEN fails
Final processing		
	BLKSIZE > LRECL, RECFM = F	RECFM altered to include blocking
	BLKSIZE > LRECL, RECFM = V	RECFM altered to include blocking
Number of buffers (BUFNO)	BUFNO ≤ 1	Multiple buffering disabled
Number of channel programs (NCP)	BUFNO ≥ 2, BUFNO > NCP	Multiple buffering enabled with BUFNO buffers
	BUFNO ≥ 2, BUFNO ≤ NCP	Multiple buffering enabled with NCP + 1 buffers
	NCP ≥ BUFNO, BUFNO ≥ 2	BUFNO = NCP + 1, multiple buffering enabled

Examples of sequential data set processing

The following examples describe various ways that Model 204 opens sequential data sets:

- Model 204 opens a dump data set that has a block size of 13000. The Model 204 page size is 6184. Therefore, the record format is set to FB, the data length and logical record length are set to 6184, and block size is rounded down to 12368 ($6184 * 2$).
- Model 204 opens the password data set (CCASTAT). No information has been provided by the operating system. Therefore, the record format is forced to be undefined. The data length, logical record length, and block size are set to 6286.
- Model 204 opens a USE data set. The operating system provides the information that the record format is FA and the logical record length is 133. The data set definition block indicates that the data length is 72. Using this information and the algorithm, the record format is set to FA, the data length is set to 72, and the logical record length and block size are set to 73.

The following examples illustrate how, after values are obtained from the operating system and the data set definition, Model 204 might force some or all the characteristics to have values dependent upon the use of the data set being opened:

- The record format for a dump data set is forced to fixed. Data length is forced to be equal to the Model 204 page size currently in effect.
- The record format for the Model 204 password data set (CCASTAT) is forced to be undefined. Data length is set to 6286, if the logical record length and block size have not been provided by the operating system.
- The record format for a USE data set is set to the value of the UDDRFM parameter, if the record format has not been provided by the operating system or a data set definition. Data length is set to 132, if data length, logical record length, and block size have not been provided by the operating system or a data set definition.
- A USE data set can be a disk file with any organization. The printer data set with a length of 133 and a record format of FA is the default.

VSAM I/O processing

External file I/O support allows READ access to any VSAM KSDS cluster through its primary or alternate index key via a predefined PATH and an ALTERNATEINDEX cluster in the VSAM catalog.

The following sections explain system requirements, primary and alternate index key processing, and procedures for loading VSAM KSDS and ESDS files.

System requirements

VSAM I/O processing requires the following system specifications:

- Increased value of the STRINGS option if users reach (and cannot reduce) the maximum number of concurrent positioning requests for a keyed VSAM data set

When the maximum number of concurrent positioning requests is reached, a request for another positioning string is denied.

- Increased dynamically allocated storage (SPCORE)

Access to VSAM data sets increases the requirements for dynamically allocated storage available at execution. SPCORE size requirements depend on:

- Location where VSAM modules are loaded

Load VSAM modules in the LPA (z/OS) or SVA (z/VSE). If the modules are not loaded into the LPA, approximately 280K bytes are required within the region or partition.

- The number of strings (STRNO) for each VSAM file and the BUFFER SPACE parameter defined in the VSAM cluster definition

The product of these parameters determines the amount of storage required for each opened VSAM file.

For z/OS installations, in particular, if the STRNO parameter is specified in the DD statement, it must have the same value as the STRINGS parameter in the DEFINE DATASET statement. Preferably, do not specify the STRNO parameter in the DD statement.

Primary key processing

When processing VSAM files through the primary key, Model 204 requires:

- DD (DLBL for z/VSE) statement in the JCL for each VSAM file
- DEFINE DATASET command in Model 204 for each VSAM file

DEFINE DATASET must contain:

- File name
- File organization (VSAM)
- Other parameters, such as access method (keyed or sequential), password (if any), number of strings, and close option

Alternate index key processing

When processing VSAM files through the alternate index key, Model 204 requires:

- DD (DLBL for z/VSE) statement in the JCL for each predetermined PATH
- DEFINE DATASET command for each PATH

Record retrieval through the alternate index key is based on the cluster definition ALTERNATEINDEX.

- If NONUNIQUEKEY is specified, the record returned from VSAM depends on the retrieval method used in the User Language procedure:
 - If a keyed retrieval (retrieval by a READ IMAGE statement with a KEY option) is requested, only the first record of the record set with the specified key is retrieved from the base cluster. Other records of the same set are ignored.
 - If sequential retrieval (a POSITION statement used to set the initial pointer, followed by READ IMAGE statement with the NEXT option) is requested, all records with the same alternate key are returned first. Records with the next highest alternate key follow.
- If UNIQUEKEY is specified in the definition of the ALTERNATEINDEX cluster, any retrieval request with the alternate key is identical to the retrieval through the primary key.

Loading a VSAM KSDS or ESDS file

A VSAM KSDS file (or a VSAM ESDS file accessed through an ALTERNATEINDEX cluster) can be loaded directly into any Model 204 file by using the File Load utility. The following specifications are required:

- Include a DD (or DLBL) statement for the VSAM file (or the PATH) in the JCL for File Load execution.
- Provide a DEFINE DATASET command for each VSAM file (or PATH definition if the alternate index is used):
 - Parameters that support VSAM file I/O are SEQUENTIAL and KEYED SEQUENTIAL file organization, to specify the VSAM access password assigned when the cluster was defined.
The password is checked against the VSAM catalog for the data set and also checked each time an OPEN DATASET is issued for the data set.
 - CLOSE indicates physically closing the VSAM file.
 - CLOSE=EOJ indicates closing the VSAM file during Model 204 termination.
 - CLOSE=NOUSERS indicates closing the VSAM file when no users are accessing the file. NOUSERS is the default.

For full details of the DEFINE DATASET command options, see the *Rocket Model 204 Parameter and Command Reference*.

- In z/VSE, load VSAM modules in the SVA area to conserve space.

22

Model 204 External Call Facility

In this chapter

- Overview
- Working with ECF
- ECF User Language statements
- ECF return codes and \$function
- ECF User 0 parameters
- Subtask and load module management
- Restrictions and cautions
- Tracking ECF
- ECF examples

Overview

The External Call Facility (ECF) is a method for programs written in Model 204 User Language to invoke external, non-Model 204 modules, such as non-IFAM COBOL modules. Data can be passed between Model 204 and external modules. External modules can open non-Model 204 data sets, read or write to them, and close them. In addition, multiple modules can be called multiple times in a single run.

ECF is available only under z/OS.

Working with ECF

Loading an external module

Each external module that is called using ECF is dynamically loaded once into memory from a nominated load library or from //STEPLIB. If Model 204 is APF authorized, then any nominated load library identified for use by ECF must also be APF authorized. Otherwise, the EXTERNAL LOAD statement will fail with:

```
COMPLETION CODE=S306 REASON CODE=000C.
```

The load is done by an authorized user executing the EXTERNAL LOAD statement; for example, by User 0 during startup. The load is issued from a dedicated ECF subtask that is used exclusively for module loads and deletes.

After an external module is loaded, a *call-name* of up to 48 characters is associated with it. An association is set up by an authorized user executing the EXTERNAL NAME statement; typically by User 0 during startup. More than one call name can be associated with the same load module name.

Callers of the external module always refer to the call (logical) name rather than the load module (physical) name. Thus, if a load module name or location changes, only the setup statements need to be changed; no application code changes are required.

Calling an external module

A previously loaded external module is called by a user executing the EXTERNAL CALL statement. This invokes a separate ECF subtask to run the external module, and causes the user to enter a swappable, bumpable wait state. When the external module completes, control returns to the user. There are no restrictions on the length of time the external module can run.

Data is passed between Model 204 and an external module via a Model 204 image. A copy of the image is made when the module is called, so that the caller need not remain in the server while the external module runs. The external module can modify the copy of the image. The modified image is returned to the caller, unless the EXTERNAL MODULE statement specified PARMTYPE=INPUT.

Avoiding data corruption and incorrect results

External modules must not access any data set that Model 204 or any other external module has already opened.

Warning: *Failure to observe this could result in data corruption or abends, either in the external module or Model 204.*

External modules invoked using ECF run under separate z/OS subtasks in the same address space as the Model 204 Online that invoked the module. This

means that external modules run in parallel with Model 204 (and each other), even if you do not use MP/204 (multiprocessor).

Most errors that can occur in the external module are isolated from Model 204. However, it is still possible for programming errors in external modules to corrupt or overwrite storage belonging to Model 204. *An programming error in your external module could result in data corruption, abends, or incorrect results in the external module or in Model 204.*

Stopping an external load module

Authorized users can stop an external module in the following ways; by executing:

- **EXTERNAL STOP** statement, usage of a load module can be disabled by an authorized user. It can be enabled again by the **EXTERNAL START** statement.
- **EXTERNAL DELETE** statement, a load module can be removed from storage by an authorized user. An **EXTERNAL DELETE** statement must not be used for Language Environment (LE) modules, because z/OS does not support the use of **DELETE** of Language Environment main programs and an abend can result.

ECF statistics and messages

See Table A-1 on page 548 for the System-final, user-logout, user-since-last, SMF-logout, and SMF-since-last statistics that relate to ECF.

BUMP command enhanced with the FORCE option for ECF users

Simply bumping a user that is executing an **EXTERNAL CALL** statement does not interrupt the external module. The external module is allowed to complete—however long that takes—and then the bump takes effect.

To interrupt an external module, you must issue a **BUMP** command with the **FORCE** option. Or, you can continue to use the **EXTERNAL STOP** statement with the **FORCE** option.

If the **FORCE** option is specified on a **BUMP** command, then the users who are running an external module are interrupted. The **FORCE** option may be combined with any other **BUMP** options. When you specify the **FORCE** option, it must be the first option on the command.

Examples

```
BUMP FORCE ALL
BUMP FORCE SUBSYS MYAPSY
BUMP FORCE MODULE MYPGM
```

For users who are not running an external module, the presence or absence of the FORCE option is irrelevant. Adding the FORCE option affects only users who are running an external module. The FORCE option does not interrupt users that are in an unbumpable wait.

See the *Rocket Model 204 Parameter and Command Reference* for more description of the BUMP command.

ECF User Language statements

You must check \$STATUS and \$STATUSD return codes after each ECF statement.

EXTERNAL CALL statement

Function Calls an external module that was previously loaded using an EXTERNAL LOAD statement, using a call name that was previously specified in an EXTERNAL NAME statement.

Privileges Any user

Syntax `EXTERNAL CALL call-name [WITH image-1
[, image-2, ... image-40]]`

Where The arguments are:

Argument	Specifies...
call-name	Logical name, either a literal or %variable, of the external module to invoke.
WITH clause	The optional WITH clause specifies the images to pass as a parameter area between the User Language program and the external module. If no parameter area is required, you can omit the WITH clause.
image-1,...image-40	Name(s) of a previously defined image(s) to pass to the external module.

Usage You can specify from one to forty images separated by commas.

Notes:

A return code of zero does *not* mean that the external module performed as it was designed; it simply means that the module was successfully invoked and completed without anabend.

Without a parameter area, the external module cannot directly pass any data back to a User Language program other than via its return code. It can,

of course, indirectly pass data back: for example, by updating a sequential data set that the User Language program can then inspect.

EXTERNAL DELETE statement

Function Removes a previously loaded external module from storage.

Privileges System manager or User 0

Syntax `EXTERNAL DELETE module-name`

Where The argument, a literal, or a %variable, is:

Argument	Specifies...
module-name	Module name of a previously loaded external module.

Usage Do not use the EXTERNAL DELETE statement for Language Environment modules.

EXTERNAL LOAD statement

Function Loads an external module into storage. The module, previously defined by an EXTERNAL MODULE statement, is loaded and is then available to all users.

Privileges System manager or User 0

Syntax `EXTERNAL LOAD module-name`

Where The argument, a literal, or a %variable, is:

Argument	Specifies...
module-name	Name of the external module, a PDS member name.

EXTERNAL MODULE statement

Function Defines an external module for later loading.

Privileges System manager or User 0

Syntax `EXTERNAL MODULE module-name [DDNAME=ddname]
 [PARMTYPE=INPUT | OUTPUT]
 [PARMSIZE=value]
 [PARMMODE=[24 | 31]]
 [REENTRANT | AFFINITY]`

Where The arguments, either literals or %variables, are:

Argument	Specifies...
<i>module-name</i>	Name of the external module, a PDS member name. Required.
<i>ddname</i>	DDNAME of the PDS where the load module is located. If omitted, the standard search order or //STEPLIB is used. Optional. Note: If Model 204 is APF authorized, then this data set must also be APF authorized.
PARMTYPE	Parameter (image) type passed to the external module on an EXTERNAL CALL statement. Optional. <ul style="list-style-type: none"> INPUT means that any changes made by the external module to the parameter are discarded. OUTPUT, the default, means that any changes are retained.
PARMSIZE	Required size of the parameter (image) passed to the external module on all EXTERNAL CALL statements. If the actual parameter size does not match the required size, the EXTERNAL CALL statement fails. If PARMSIZE is not specified, the parameter size is not checked.
PARMMODE	Where storage to hold the copy of the parameter area is allocated. Optional. <ul style="list-style-type: none"> 24 means allocate in 24-bit, below-the-line storage. 31, the default, means allocate in 31-bit, above the line storage. Note: Do not specify PARMMODE=24 unless the module has a specific requirement for this setting.
REENTRANT	Module that can be used by more than one user at a time. This optional argument takes effect only if the module was link-edited with the REENTRANT attribute.
AFFINITY	Module always runs on the same subtask. Conversely, that subtask runs only that module.

Usage You can use an EXTERNAL MODULE statement to define a module using either:

- New name and attributes.
- Existing name. Attributes associated with an existing name are replaced by new attributes.

Subtask affinity, which is specified by the AFFINITY keyword, is required in some situations. For example, when an external module opens a data set on one call, but does not close it till a subsequent call. Subtask affinity is required

because z/OS requires that you open and close a data set from the same subtask.

The AFFINITY keyword is incompatible with the REENTRANT keyword. A compilation error is generated, if both are specified for the same module. You specify one or the other or neither, but not both.

EXTERNAL NAME statement

Function Use the EXTERNAL NAME statement to:

- Associate a logical call name with the name of a module that was previously defined by issuing an EXTERNAL MODULE statement
- Remove a previously set up association. A module can have multiple call names.

Privileges System manager or User 0

Syntax

```
EXTERNAL NAME call-name FOR module-name
EXTERNAL NAME call-name REMOVE
```

Where The arguments, either literals or %variables, are:

Argument	Specifies...
<i>call-name</i>	Logical name to associate with an external module. Up to 48 characters.
<i>module-name</i>	Name of a previously loaded external module.

EXTERNAL START statement

Function Enables further calls to an ECF module.

Privileges System manager or User 0

Syntax

```
EXTERNAL START module-name
```

Where The argument, either literals or %variables, is:

Argument	Specifies...
<i>module-name</i>	Name of a previously loaded external module.

Usage Initially a module is in the START state. You need not issue an EXTERNAL START statement unless you want to reverse a prior EXTERNAL STOP statement.

EXTERNAL STOP statement

Function Stops further calls to an ECF module. Currently executing calls either complete or, if the FORCE option is used, abend.

Privileges System manager or User 0

Syntax `EXTERNAL STOP module-name [FORCE]`

Where The arguments, which can be either literals or %variables, are:

Argument	Specifies...
module-name	Name of a previously loaded external module.
FORCE	Users currently executing the specified module are bumped.

Usage The following table describes how an EXTERNAL STOP command is evaluated.

When EXTERNAL STOP command is issued	Then
If no users are executing the module	Module is immediately marked <i>stopped</i> .
If one or more users are executing the module	Module is marked <i>draining</i> until the last user finishes executing the module, then it is marked <i>stopped</i> .
If FORCE was specified	Current users of the module are bumped.
If FORCE was not specified	Current users of the module are allowed to complete.

In all cases, subsequent attempts by any user to call the module with an EXTERNAL CALL statement result in a \$STATUS of 8.

Note: Rocket Software *does not recommend* using the FORCE option, as Model 204 cannot ensure that the external module is terminated cleanly.

ECF return codes and \$function

All EXTERNAL statements set \$STATUS and \$STATUSD.

Table 22-1. ECF return codes

\$STATUS	\$STATUSD	Meaning
0	0	ECF function completed without error
1	0	ECF inactive

Table 22-1. ECF return codes (Continued)

\$STATUS	\$STATUSD	Meaning
2	0	Not authorized
3	1	Invalid module name
	2	Invalid call name
	3	Invalid DDNAME
	4	Image inactive
	5	Invalid PARMSIZE
4	0	Call name not defined
6	0	Module not defined
7	0	Module not loaded
8	0	Module unavailable (draining or stopped)
10	0	Module not deleted
20		System busy; timed out
	1	Module unavailable
	2	No subtask available
30		Load or delete failed
	1	DDNAME not present
	2	DDNAME open failed
	3	DDNAME close failed
	5	Load or delete failed; see the \$ECFSTAT function in the <i>Rocket Model 204 User Language Manual</i> in the chapter on \$functions.
	6	Internal ECFabend; see the \$ECFSTAT function in the <i>Rocket Model 204 User Language Manual</i> in the chapter on \$functions.
	7	Internal ECF ABEND; load or delete subtask terminated by the operating system.
40	0	Module failed
	1	Module gave nonzero return code; see the \$ECFSTAT function in the <i>Rocket Model 204 User Language Manual</i> in the chapter on \$functions.
	2	Moduleabend; see the \$ECFSTAT function in the <i>Rocket Model 204 User Language Manual</i> in the chapter on \$functions.

Table 22-1. ECF return codes (Continued)

\$STATUS	\$STATUSD	Meaning
	3	Insufficient memory available to allocate a buffer for the parameter area
	4	Actual parameter size is not equal to PARMSIZE
	5	When the parameter area, as updated by the external module, was being copied back to the original image(s), ECF detected that the size of one of the images had been changed. This status can occur only if an image contains a variable array whose size is changed by the external module.
	6	ECF subtask terminated by the operating system
50	0	ECF internal table full
	1	ECF ECMODS table full; increase ECMODS User 0 parameter
	2	ECF ECNAMES table full; increase ECNAMES User 0 parameter

\$ECFSTAT function

The \$ECFSTAT function returns the detailed completion code from the previous EXTERNAL statement. See the *Rocket Model 204 User Language Manual* in the chapter on User Language functions.

ECF User 0 parameters

The following User 0 parameters are used by ECF.

A system manager can reset only the ECWAIT parameter; the other ECF parameters cannot be reset.

User 0 parameter	Returns...
ECISUBS	Subtasks for running external modules initially attached
ECMODS	External modules to load
ECMSUBS	Number of ECF subtasks for running modules
ECNAMES	Number of external call names
ECPRIV	ECF privileges
ECPSIZE	Size of largest image used by ECF
ECWAIT	ECF wait time

See the *Rocket Model 204 Parameter and Command Reference* in the chapter on parameters A–F for more details about the parameters that are used by ECF.

Subtask and load module management

Some z/OS overhead accrues in loading an external module into storage and attaching a z/OS subtask under which it runs. ECF avoids incurring this cost for every EXTERNAL CALL statement by managing the load modules and subtasks as described in the following sections.

Subtasks assignment

At system startup, the following subtasks are started:

- One subtask for loading and deleting modules
- One or more subtasks, specified by the ECISUBS parameter, for executing external modules

When a user issues an EXTERNAL CALL statement, an unused ECF subtask is selected on which to run the external module, up to the limit of ECMSUBS. Users unable to get a subtask enter a wait of up to ECWAIT milliseconds for a subtask to become available. When an in-use ECF subtask becomes available, because the module that was running under it ends, it is assigned to a user waiting for a subtask. If a user's wait time expires before a subtask is available, a no-subtask-available failure is returned.

Subtask affinity

Normally, when an external module is called via the EXTERNAL CALL statement, ECF selects any free subtask on which to execute the module. Although this is appropriate for most external modules, some modules might need to always be executed on the same subtask. This is known as **subtask affinity**. Subtask affinity for a module is specified by the AFFINITY option on the EXTERNAL MODULE statement. The AFFINITY option is incompatible with the REENTRANT option.

- If the AFFINITY option is not specified, the module does not have subtask affinity. This is the default.
- If the AFFINITY option is specified, the module has subtask affinity.

When the first EXTERNAL CALL of the module is executed, a dedicated subtask for the module is attached. A two-way association between the module and the subtask is established. That module will run on only that subtask, not on any other subtask. That subtask is used to run only that module, not any other module.

If a module has subtask affinity, and the dedicated subtask for that module is abnormally terminated for any reason, the previous dedicated subtask is detached, and a new dedicated subtask for the module is attached. A subtask

could be abnormally terminated if the module it was running has an abend, or the user running the module is bumped. Therefore, specifying AFFINITY does not guarantee that the same subtask is always used for that module for the duration of a Model 204 Online.

If you use modules with subtask affinity, ensure that ECISUBS and ECMSUBS are appropriately set. In particular, ECISUBS and ECMSUBS must be greater than or equal to the number of modules with subtask affinity.

- The ECISUBS parameter specifies the number of subtasks attached during system initialization. These subtasks are never used for modules with subtask affinity; the first EXTERNAL CALL of a module with subtask affinity always results in a new subtask being attached (subject to ECMSUBS).
- The ECMSUBS parameter specifies the maximum number of ECF subtasks used to run modules. The number of subtasks used for modules with subtask affinity *plus* the number of subtasks used for modules without subtask affinity will be a maximum of ECMSUBS.

Load modules

An external module is usually loaded into storage just once. The exception is when an EXTERNAL LOAD statement, possibly preceded by an EXTERNAL DELETE statement, is used to reload a previously-loaded module.

Normally, one Model 204 user at a time is allowed to issue an EXTERNAL CALL statement for the module. Other users who attempt to call the module while it is in use, enter a wait-state of up to ECWAIT milliseconds for the module to become available. When a module is freed, because the current execution of it ends, it is assigned to one of the users waiting for it. If a user's wait time expires before the module is assigned, a module-in-use failure is returned.

If a load module was link-edited with the REENTRANT (also written, RENT) attribute, and if the EXTERNAL MODULE statement that defined the module characteristics specified the REENTRANT option, then multiple Model 204 users are allowed to simultaneously issue an EXTERNAL CALL statement for the module—subject to subtask availability.

Fulfillment order

An EXTERNAL CALL statement can invoke an external module only if both an ECF subtask is free, and for a serially reusable module, if the module is not in use by another user. The check that the module is not in use by another user is done before the allocation of a subtask so that the most restrictive condition is checked first.

Restrictions and cautions

The following restrictions and cautions apply to the use of ECF:

- ECF loads an external module only once, when the EXTERNAL LOAD statement is issued. Thereafter, every user who issues an EXTERNAL CALL statement for that module uses the same copy of the module. Therefore, you must write your module to initialize itself properly on every call.
- ECF passes the copy of the parameter area to the external module using standard z/OS linkage conventions. Your external module must support these conventions to receive the parameters.
- ECF does not provide any special initialization of the environment when you issue an EXTERNAL CALL statement; it merely branches to the in-memory copy of the module. Therefore, your module must perform any required initialization and termination of its run-time environment. In particular, this means that an external module, written in any language, should be written as a main routine and not a subroutine.
- Only one copy of a load module can be in memory at a time. This is a z/OS restriction.
- Externally called modules must be AMODE(31) and either RMODE(ANY) or RMODE(31).
- Externally called modules must not attempt to retain any context information from one call to another. You must write the modules so that each call executes independently of any other.
- For efficiency purposes, ECF does not use z/OS to maintain the usage counts or status information for load modules. Control is passed by direct branch rather than use of the z/OS ATTACH, LINK, or XCTL macros. Therefore, one external module must not attempt to load or attach another external module, or attempt to reference code or data in another external module.
- The definition of the parameter area in Model 204—the image definition—must agree with the definition of the parameter area in the external module (in a COBOL program, the LINKAGE SECTION). If they do not agree, it is possible for the module to modify the wrong storage.

To prevent this, Model 204 checks that the external module put its results in *only* the assigned area—not somewhere else. Model 204 checks the area immediately past the end of the assigned area. If the unassigned area was used, Model 204 displays the following message and restarts the user.

```
M204.2563: MODULE=name RETURNED MORE THAN length BYTES
```

Tracking ECF

Wait types for ECF

See Table 4-6 on page 134 for a listing of wait types that includes the External Call Facility, wait types 43 through 46.

ECF statistics

The following statistics help you track the External Call Facility. See Table A-1 on page 548 for the position in the system-final, user-logout, user-since-last, SMF-logout, and/or SMF-since-last journal record layout as they apply.

- ECCALL
- ECCNCT
- ECCTOUT
- ECCWAITM
- ECCWAITS
- ECDELETE
- ECLOAD
- ECMODMAX
- ECNAMMAX
- ECTSKMAX
- ECTWAITM
- ECTWAITS

Statistics include only EXTERNAL CALL statements that actually called a module—even if the module subsequently abended. ECF statistics do not include EXTERNAL CALL statements with parameter errors or those that timed out trying to get a module or subtask.

ECF examples

This section illustrates, in various languages, how to write, compile, link and invoke an external module that adds two numbers together and returns the sum. For clarity, the sample code omits error handling, other standard elements, and some JCL elements.

COBOL sample Number 1

This example uses the Language Environment enabled compiler, IBM COBOL FOR z/OS AND z/VM.

COBOL program

```
//COBSAMP EXEC PGM=IGYCRCTL, PARM=(NOSEQ, RENT)
//SYSLIN DD DSN=YOUR.OBJLIB(COBSAMP), DISP=SHR
//SYSIN DD *
IDENTIFICATION DIVISION.
```

```

PROGRAM-ID. COBSAMP.
DATA DIVISION.
LINKAGE SECTION.
01  M204-PARMS.
    03  NUMBER-ONE          PIC S9(7)  COMP-3.
    03  NUMBER-TWO          PIC S9(9)  BINARY.
    03  NUMBER-SUM          PIC S9(9)  BINARY.
PROCEDURE DIVISION USING M204-PARMS.
    COMPUTE NUMBER-SUM EQUAL NUMBER-ONE + NUMBER-TWO
    MOVE ZERO TO RETURN-CODE
    GOBACK.

```

Language Environment Options

```

//ASMUOPT EXEC PGM=ASMA90,PARM='NOXREF'
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
// DD DSN=CEE.SCEEMAC,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIN DD DSN=YOUR.OBJLIB(CEEUOPT),DISP=SHR
//SYSIN DD *
CEEUOPT CSECT
CEEUOPT AMODE ANY
CEEUOPT RMODE ANY
PRINT ON,NOGEN
CEEUOPT
ABTERMENC=(ABEND),
RTEREUS=(ON)
END

```

Linkedit

```

//LINK EXEC PGM=IEWL,PARM='LIST,MAP'
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=CEE.SCEELKED,DISP=SHR
//OBJLIB DD DSN=YOUR.OBJLIB,DISP=SHR
//SYSLMOD DD DSN=YOUR.LOADLIB,DISP=SHR
//SYSLIN DD *
INCLUDE OBJLIB(CEEUOPT)
INCLUDE OBJLIB(COBSAMP)
NAME COBSAMP(R)

```

Calling from User Language

```

//ONLINE EXEC PGM=ONLINE
//STEPLIB DD DSN=M204.LOADLIB,DISP=SHR
// DD DSN=YOUR.LOADLIB,DISP=SHR
// DD DSN=CEE.SCEERUN,DISP=SHR

```

```

BEGIN
IMAGE PARMS
    NUMBER.ONE.PACKED IS PACKED LEN 4
    NUMBER.TWO.BINARY IS BINARY LEN 4
    NUMBER.SUM.BINARY IS BINARY LEN 4
END IMAGE

EXTERNAL MODULE COBSAMP
EXTERNAL LOAD COBSAMP
EXTERNAL NAME MOD_COBSAMP FOR COBSAMP

PREPARE IMAGE PARMS
%PARMS:NUMBER.ONE.PACKED = 1
%PARMS:NUMBER.TWO.BINARY = 2
EXTERNAL CALL MOD_COBSAMP WITH PARMS
PRINT %PARMS:NUMBER.ONE.PACKED WITH ' + ' WITH -
      %PARMS:NUMBER.TWO.BINARY WITH ' = ' WITH -
      %PARMS:NUMBER.SUM.BINARY
END

```

Comments.

COBOL	Corresponds to User Language
PIC S9(7) COMP-3	PACKED LEN 4
PIC S9(9) BINARY	BINARY LEN 4

- The Language Environment option ABTERMENC= (ABEND) *must* be specified.
- Specify the Language Environment option RTEREUS= (ON) and code GOBACK instead of STOP RUN, as illustrated, to make the runtime environment reusable and to improve performance.
- Setting Language Environment options is described in IBM Manual SC28-1939 *Language Environment for z/OS & z/VM Programing Guide*. In the code in the previous section, “Calling from User Language”, Language Environment options are set by linking CEEUOPT with the COBOL module. Other methods are also available; check with your site’s Language Environment administrator to determine the appropriate method to use.

COBOL sample Number 2

This example illustrates the use of multiple images using the Language Environment enabled compiler, IBM COBOL FOR z/OS AND z/VM.

Note: Ordinarily, multiple images are only used if the parameters to be passed cannot fit into a single image. Images are limited in size to 32767 bytes.

COBOL program

```

IDENTIFICATION DIVISION.
PROGRAM-ID. COBSAM2.
DATA DIVISION.
LINKAGE SECTION.
01  M204-PARMS1.
    03  NUMBER-ONE                PIC S9(7) COMP-3.
01  M204-PARMS2.
    03  NUMBER-TWO                PIC S9(9) BINARY.
01  M204-PARMS3.
    03  NUMBER-SUM                PIC S9(9) BINARY.
PROCEDURE DIVISION USING M204-PARMS1, M204-PARMS2, M204-PARMS3.
    COMPUTE NUMBER-SUM EQUAL NUMBER-ONE + NUMBER-TWO
    MOVE ZERO TO RETURN-CODE
    GOBACK.

```

Calling from User Language

```

BEGIN
IMAGE PARMS1
    NUMBER.ONE.PACKED IS PACKED LEN 4
END IMAGE
IMAGE PARMS2
    NUMBER.TWO.BINARY IS BINARY LEN 4
END IMAGE
IMAGE PARMS3
    NUMBER.SUM.BINARY IS BINARY LEN 4
END IMAGE

EXTERNAL MODULE COBSAM2
EXTERNAL LOAD COBSAM2
EXTERNAL NAME MOD_COBSAM2  FOR COBSAM2

PREPARE IMAGE PARMS1
PREPARE IMAGE PARMS2
PREPARE IMAGE PARMS3
%PARMS1:NUMBER.ONE.PACKED = 1
%PARMS2:NUMBER.TWO.BINARY = 2
EXTERNAL CALL MOD_COBSAM2 WITH PARMS1, PARMS2, PARMS3
PRINT %PARMS1:NUMBER.ONE.PACKED WITH ' + ' WITH -
      %PARMS2:NUMBER.TWO.BINARY WITH ' = ' WITH -
      %PARMS3:NUMBER.SUM.BINARY
END

```

SAS/C sample

This example uses the SAS/C compiler, not the IBM C compiler.

SAS/C program

```
//C          EXEC PGM=LC370B
//STEPLIB   DD   DSN=SASC.LOAD,DISP=SHR
//SYSLIB    DD   DSN=SASC.MACLIBC,DISP=SHR
//SYSLIN    DD   DSN=YOUR.OBJLIB(SASCSAMP),DISP=SHR
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD   *
typedef struct PARMAREA {
    int A;
    int B;
    int C;
} parmarea;

int main(int argc, char **argv) {
    parmarea *pptr;
    if (argc != 2) {
        /* the parameter list was not in OS format */
        return 1000;
    }
    pptr = (parmarea *) argv[1];
    pptr->C = pptr->A + pptr->B;
    return 0;
}
```

Linkedit

```
//LC          EXEC PGM=IEWL, PARM='AMODE(31),RMODE(ANY)'
//SYSLIB      DD   DSN=SASC.STDLIB,DISP=SHR
//            DD   DSN=SASC.BASELIB,DISP=SHR
//OBJLIB      DD   DSN=YOUR.OBJLIB,DISP=SHR
//SYSLMOD     DD   DSN=YOUR.LOADLIB,DISP=SHR
//SYSPRINT    DD   SYSOUT=*
//SYSLIN     DD   *
    INCLUDE   OBJLIB(SASCSAMP)
    ENTRY     $MAINC
    NAME      SASCSAMP(R)
//*
```

Calling from User Language

```
//ONLINE     EXEC PGM=ONLINE
//STEPLIB    DD   DSN=M204.LOADLIB,DISP=SHR
//           DD   DSN=YOUR.LOADLIB,DISP=SHR
```

```

BEGIN
IMAGE PARMS
    NUMBER.ONE.BINARY IS BINARY LEN 4
    NUMBER.TWO.BINARY IS BINARY LEN 4

    NUMBER.SUM.BINARY IS BINARY LEN 4
END IMAGE

EXTERNAL MODULE SASCSAMP
EXTERNAL LOAD SASCSAMP
EXTERNAL NAME MOD_SASCSAMP FOR SASCSAMP

PREPARE IMAGE PARMS
%PARMS:NUMBER.ONE.BINARY = 1
%PARMS:NUMBER.TWO.BINARY = 2
EXTERNAL CALL MOD_SASCSAMP WITH PARMS
PRINT %PARMS:NUMBER.ONE.BINARY WITH ' + ' WITH -
      %PARMS:NUMBER.TWO.BINARY WITH ' = ' WITH -
      %PARMS:NUMBER.SUM.BINARY
END

```

Comments.

SAS/C	Corresponds to User Language
int	BINARY LEN 4

- The SAS/C program must be coded as a main program, not a subroutine.
- To enable SAS/C to accept parameters in standard OS format, the entry point must be defined as \$MAINC and the parameters handled as illustrated. This technique is described in the *SAS/C Compiler and Library User's Guide*.

Assembler sample

This example uses the non-Language Environment assembler.

Assembler program

```

//ASMSAMP EXEC PGM=ASMA90
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSLIN DD DSN=YOUR.OBJLIB(ASMSAMP),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ASMSAMP CSECT

ASMSAMP AMODE 31
ASMSAMP RMODE ANY

```

ECF examples

```
        USING *,15
        L      1,0(1)      GET ADDRESS OF IMAGE COPY
        XC      WORKAREA,WORKAREA      CLEAR WORK AREA
        MVC      WORKAREA+4(4),0(1)      GET FIRST NUMBER (PACKED)
        CVB      0,WORKAREA      GET FIRST NUMBER (BINARY)
        A      0,4(1)      ADD SECOND NUMBER (BINARY)
        ST      0,8(1)      STORE SUM (BINARY)
        XR      15,15      ZERO RETURN CODE
        BR      14      RETURN
        DS      0D
WORKAREA DS      PL8      WORKING STORAGE
        END      ASMSAMP
```

Linkedit

```
//LASM      EXEC PGM=IEWL
//OBJLIB    DD  DSN=YOUR.OBJLIB,DISP=SHR
//SYSLMOD   DD  DSN=YOUR.LOADLIB,DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//SYSLIN    DD  *
        INCLUDE OBJLIB(ASMSAMP)
        NAME ASMSAMP(R)
```

Calling from User Language

```
//ONLINE    EXEC PGM=ONLINE
//STEPLIB   DD  DSN=M204.LOADLIB,DISP=SHR
//          DD  DSN=YOUR.LOADLIB,DISP=SHR
```

```
BEGIN
IMAGE PARMS
    NUMBER.ONE.PACKED IS PACKED LEN 4
    NUMBER.TWO.BINARY IS BINARY LEN 4
    NUMBER.SUM.BINARY IS BINARY LEN 4
END IMAGE
```

```
EXTERNAL MODULE ASMSAMP
EXTERNAL LOAD ASMSAMP
EXTERNAL NAME MOD_ASMSAMP FOR ASMSAMP
```

```
PREPARE IMAGE PARMS
%PARMS:NUMBER.ONE.PACKED = 1
%PARMS:NUMBER.TWO.BINARY = 2
EXTERNAL CALL MOD_ASMSAMP WITH PARMS
CALL CHECK.STATUS
PRINT %PARMS:NUMBER.ONE.PACKED WITH ' + ' WITH -
      %PARMS:NUMBER.TWO.BINARY WITH ' = ' WITH -
      %PARMS:NUMBER.SUM.BINARY
```

END

Comments

Assembler	Corresponds to User Language
DS PL4	PACKED LEN 4
DS F	BINARY LEN 4

23

Using Program Communication Facilities

In this chapter

- Overview
- Preparing for TPROCESS communication
- Preparing for Transfer Control communication

Overview

Program Communication, available on CICS, CMS, and SNA Communications Server (formerly VTAM) full-screen threads (IODEV=11, IODEV=41, and IODEV=7, respectively), allows User Language requests to communicate with user-written CICS programs, CMS EXECs, or SNA Communications Server applications. The Program Communication facilities, that is, TPROCESS (Terminal Process) and Transfer Control, allow a User Language request to do the following:

- TPROCESS — Identify, trigger execution of, and communicate with a CICS partner program or CMS partner EXEC
- Transfer Control — Identify a CICS partner program or SNA Communications Server partner application, transfer control to the program or application, and pass it a parameter area

In addition, Model 204 provides program-to-program processing support through its distributed application facility, Horizon, an optional feature. Model 204 applications can communicate through SNA Communications Server with one or more other programs using program-to-program processing. Communication takes place over an SNA network, using verbs and protocols conforming to LU 6.2 architecture. Horizon uses User Language statements and Model 204 commands, and might require IODEV=27 threads.

This chapter provides a summary of system management information for the support of the Program Communication facilities. Horizon is described in *Model 204 Horizon: Intersystem Processing Guide*; IODEV=27 threads are discussed in “Horizon (IODEV=27)” on page 96.

Preparing for TPROCESS communication

TPROCESS communication works under CICS or CMS. User Language requests communicate with a CICS program or CMS EXEC by issuing the OPEN PROCESS and CLOSE PROCESS commands and using SEND, RECEIVE, FLUSH PROCESS, and SIGNAL PROCESS statements.

The calls used by CICS and CMS are the following:

- CICS communicates with the User Language request by calling the IFCSA, IFSEND, IFRECV, and IFSGNL functions resolved by the IFPPCI module.
- CMS EXEC communicates with the User Language request through the IFSENDX, IFRECV, and IFSGNL subcommands.

The following sections describe the system manager actions required to enable TPROCESS. The operation of the TPROCESS and Transfer Control facilities is described in the *Model 204 User Language Manual*.

Defining the partners

Before a User Language request can communicate with either CICS or CMS, you must define a partner process (CICS program or CMS EXEC) for reference by User Language statements. Define the process and its attributes with the DEFINE LINK, DEFINE PROCESSGROUP, and DEFINE PROCESS commands. These commands connect the process to a link, identify the communication protocol to be used with the partner process, and define the partner process you must enable the link defined by the DEFINE LINK command with the OPEN LINK command.

Syntax and examples of these commands are presented in the *Model 204 Parameter and Command Reference*.

Preparing for CICS TPROCESS

Linking to IFPPCI

You must link the CICS program identified by the DEFINE command with the IFPPCI module.

IFPPCI has five entry points (IFCSA, IFPREP, IFSEND, IFRECV, and IFSGNL) that the CICS program can call. These calls are described in the *Model 204 User Language Manual*.

Installing the full-screen CICS Interface

The PPT entry for the full screen CICS Interface module name (M204PSFS) must specify a TWA size.

Specifying buffer sizes

Screen and CRAM buffer sizes are dependent on the model type and LOU TPB size specification. The LOU TPB parameter for IO DEV=11 determines the CRAM buffer size. Set LOU TPB slightly larger than the screen size for the 3270 model type used. If users are resetting model types, set LOU TPB to match the largest screen size.

Preparing for CMS TPROCESS

Installing the M204USR module

In CMS, the reentrant M204USR module, which must be installed as a saved segment or as a nucleus extension (see the *Rocket Model 204 z/VM Installation Guide*), acts as the interface that calls a CMS EXEC (REXX or EXEC2) to communicate with a User Language request through subcommands. M204USR is the module name, if the M204USR module is installed as a saved segment. M204USR is the module name, if the M204USR module is installed as a nucleus extension.

Installing M204USR as a saved segment or nucleus extension avoids the possibility of overlaying the interface module by commands that run in the user area.

Constraints

Model 204 supports communication between a User Language request and one concurrent CMS partner process, subject to the following constraints:

- The CMS partner process must be written in System Product Interpreter language (REXX) or EXEC2. Communication occurs through the standard CMS SUBCOM interface, which is supported by these languages.

- If any program executed by the CMS partner process terminates abnormally, the user's thread is disconnected.

For more information

Functions and subcommands used in TPROCESS are described in the *Model 204 User Language Manual*. Syntax and examples of these commands are presented in the *Model 204 Parameter and Command Reference*. User Language statements are described in the *Model 204 User Language Manual*.

Preparing for Transfer Control communication

In Transfer Control, User Language requests transfer control to a CICS program or SNA Communications Server application by issuing the TRANSFER statement. Control is passed to the CICS or SNA Communications Server application partner process, and the CICS or SNA Communications Server application user is disconnected from Model 204.

This section presents overview information for the system manager about the Transfer Control facility. The primary source for operational information for the facility is the *Model 204 User Language Manual*.

SNA Communications Server transfers are more complex

CICS transfers are within a single region (between the Model 204 CICS Interface and a CICS program running in the same region). SNA Communications Server transfers are between SNA Communications Server regions, using the PASS option of the CLSDST macro. SNA Communications Server transfers are more complex:

- Inter-regional security concerns might require a transfer to provide for login and password processing at the destination region.

DEFINE command and TRANSFER statement options available only for SNA Communications Server transfers determine how the command string passed in the transfer is processed at the destination.

- You might have to specially format the passed command-line data to suit a non-Model 204 destination application.

When the destination application is another Model 204 region, the login string can be automatically formatted by specifying the appropriate DEFINE PROCESSGROUP option. In this case, the login string is also encrypted; it is decrypted at the destination Model 204 region.

- A CICS transfer failure leaves the user in the CICS Interface, whereas a SNA Communications Server transfer failure can leave the user in SNA Communications Server between applications.

A user can successfully transfer via SNA Communications Server to an application in another region but fail login requirements at the destination application. A mechanism is required to return the user from SNA

Communications Server to the original Model 204 environment or to another SNA Communications Server application.

The \$REMOTE function returns the VTAMNAME parameter value of the region from which the SNA Communications Server transfer issues. This facilitates the transfer of the user back to the initial Model 204 region.

- If a SNA Communications Server TRANSFER fails, Model 204 attempts to get the terminal back by issuing an OPNDST with the ACQUIRE option. Therefore, if the Online is to support SNA Communications Server TRANSFER, the VTAMNAME APPL definition must contain AUTH=(PASS,ACQ).

Defining the partners

Before a User Language request can transfer control to either CICS or SNA Communications Server, you must define a partner process (CICS program or SNA Communications Server application) for reference by User Language statements. Define the partner and its attributes with the DEFINE LINK, DEFINE PROCESSGROUP, and DEFINE PROCESS commands. These commands connect the process to a link, identify the communication protocol to be used with the partner process, and define the partner process.

In addition, the link defined by the DEFINE LINK command must be enabled with the OPEN LINK command.

Syntax and examples of these commands are presented in the *Model 204 Parameter and Command Reference*.

Transferring to CICS

The CICS partner process must be a CICS program defined to CICS.

After the User Language TRANSFER statement completes successfully, the user is disconnected from Model 204 and transferred to the specified CICS partner process.

Data passed to CICS via the TRANSFER statement PASSING clause is written to a Temporary Storage Queue named by combining the CICS user's terminal ID and the string PSFS. The CICS partner reads the data from the Temporary Storage Queue and processing continues according to the CICS program.

Transferring to SNA Communications Server

After the User Language request containing the TRANSFER statement transfers the user to the destination region, and a session is established with the destination SNA Communications Server application, the user is notified that the session is established and is disconnected from Model 204. Data passed in the transfer (for example, Model 204 commands) is acted upon in the destination application.

If SNA Communications Server fails to establish a session with the destination application, the user is notified that the session is not established. The user is not disconnected from Model 204, and control returns to the request that contains the TRANSFER statement.

The application transferred to is responsible for transferring the user back to the original region or to another SNA Communications Server application.

Proper login and handling of the passed command string requires coordination between the security options specified in the DEFINE commands and the TRANSFER statement. Selection of these security options is discussed in the *Model 204 User Language Manual*.

For more information

Syntax and examples of the DEFINE and OPEN LINK commands are presented in the *Model 204 Parameter and Command Reference*. The User Language TRANSFER statement and \$REMOTE function are described in the *Model 204 User Language Manual*.

A

Using System Statistics

In this chapter

- Overview
- Description of statistics
- System Management Facility record layout and statistics
- Header and trailer entries (Type 0)
- Recovery entries (Types 1–6)
- System statistics entries (Type 8)
- System final and partial statistics
- System performance statistics
- Additional disk buffer monitor statistics
- Multiprocessing (MP) subtask statistics
- User statistics entries (Type 9)
- User final and partial statistics
- User since-last statistics
- User performance statistics
- File statistics entries (Type 10)
- Text entries (Type 11, Type 13)

- Initialization entries (Type 12 — X'0C')
- Time-stamp entries (Type 14 — X'0E')
- Merged journal bracketing entries (Type 15)

Overview

The tables in this appendix are provided for your use in identifying and tracking statistics that are reported by Model 204 in various situations. The offsets, given in decimal and hexadecimal, are to help you locate particular statistics within statistical output.

The statistics are collected in the CCAJRNL data set, or if allocated, in CCAJLOG. The offsets in the tables below locate each statistic in the corresponding record. To accommodate sites that need to run Model 204 around the clock for many days at a time, space for most Model 204 statistics is a double word.

Description of statistics

Table A-1 lists the statistics that are collected and that you can evaluate. It also describes the purpose of each statistic followed by the circumstances when you can collect the information and an offset in the journal record.

To collect several statistics you must preset a parameter to a nonzero value. Statistics that require a parameter are also noted in Table A-1. Table 13-3 on page 303 lists those statistics with the required parameter.

Tables, identified by type, later in this chapter, describe the layouts of the various journal records.

Table A-1. Statistics with descriptions

Statistic	Records...
APSYLD	Number of APSY loads
APSYLDD	Number of APSY loads from a dataspace
APSYLDT	Number of tiny APSY loads
AUDIT	Number of lines written to the journal and/or the audit trail
BACKOUTS	Number of BACKOUTs
BADD	Number of fields added to Table B
BCHG	Number of fields changed in Table B
BDEL	Number of fields deleted from Table B
BLKCFRE	Number of times user held a critical file resource (CFR) exclusively and forced another user to wait Parameter setting required: see Table 13-3 on page 303.

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
BLKI	Percentage of performance samples at which the user was in a server but blocked from running and waiting for an external event to occur (reported only if the performance subtask is active) Parameter setting required: see Table 13-3 on page 303.
BLKO	Percentage of performance samples at which the user was swapped out to the server data set but blocked from running and waiting for an external event to occur (reported only if the performance subtask is active) Parameter setting required: see Table 13-3 on page 303.
BLKRLK	Number of times user was the first user identified as blocking another from obtaining a record lock Parameter setting required: see Table 13-3 on page 303.
BXCHNG	Number of value entries changed in the ordered index. This occurs when a record number is added to or removed from an existing value entry.
BXDELE	Number of value entries deleted from the ordered index
BXFIND	Number of ordered index searches to locate field name=value pairs in the ordered index
BXFREE	Number of ordered index pages (nodes) emptied and released from the ordered index
BXINSE	Number of new value entries inserted into the ordered index
BXNEXT	Number of value entries touched in the ordered index during a range retrieval
BXRFND	Number of times the cursor was repositioned in the ordered index during a search
BXSPLI	Number of Table D pages (nodes) split in the ordered index
CDLWAIT	Number of constraint dependency lock waits
CNCT	Elapsed time in seconds
COMMITs	Number of COMMITs
CPU	Total CPU time consumed in milliseconds. The system CPU statistics include both maintask and subtask CPU usage.
DEQ	Total number of units of work taken by the MAINTASK and each MP and zIIP subtask, regardless of the queue that the work was taken from. Reported at Model 204 termination or in MONITOR TASKS output. (type Subtask)

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
DEV5	Output lines to procedures defined during the run
DEV6	Input lines from included procedures
DEV7	Output lines to SNA Communications Server (formerly VTAM) 3270s
DEV8	Input lines from SNA Communications Server 3270s
DEV9	Page headers or trailers defined in requests
DEV10	Output lines written to directed output (USE) data sets
DEV11	Output lines to remote User Language full screen
DEV12	Input lines from a remote User Language full screen
DEV13	Output lines to remote SQL
DEV14	Input lines from remote SQL
DEV17	Output lines to SQL CRAM IUCU
DEV18	Input lines from SQL CRAM
DEV19	Output lines to remote SQL LU62
DEV20	Input lines from remote SQL LU62
DEV23	Input lines to CRAM IFAM
DEV24	Input lines from CRAM IUCU thread
DEV27	Terminal error messages issued for inbound Distributed Application Facility conversations
DEV28	Reserved
DEV31	IFDISP output lines to IFAM1 programs
DEV32	Input arguments supplied by certain IFAM1 calls
DEV37	Output lines to SNA Communications Server 2741s and Teletypes
DEV38	Input lines from SNA Communications Server 2741s and Teletypes
DEV49	Reserved
DEV50	Lines sent to the printer when ROUTER=CICS
DEV53	Output lines from an IFAM1 application
DEV54	Input lines from an IFAM1 application
DEV55	Reserved
DEV56	Reserved

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
DEV57	Reserved
DEV58	Reserved
DEV59	Reserved
DEV60	Reserved
DEV61	Reserved
DEV62	Reserved
DEV63	Reserved
DEV64	Reserved
DEV65	Reserved
DEV66	Reserved
DEV67	Reserved
DEV68	Reserved
DEV69	Reserved
DEV70	Reserved
DEV71	Reserved
DEV72	Reserved
DEV73	Reserved
DEV74	Reserved
DIRRCD	Number of records searched in direct searches of Table B
DKAR	Number of buffers allocated without page read requests
DKPR	Number of requests for a page (may not require real I/O)
DKPRF	Number of fast logical page reads. Number of DKPRs that were satisfied by pending or deferred close buffers
DKRD	Number of physical page reads from Model 204 files
DKRDL	Number of physical page reads into the below the bar buffer pool
DKRR	Number of real disk reads for recently requested pages Parameter setting required: see Table 13-3 on page 303.
DKSAWB	Anticipatory writes from the bottom of the LRU queue. The page in the buffer is then deleted and no longer available without a re-read. Usually a small statistic and typically only incremented by a few rare events. Offset: see Table A-7 on page 572.

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
DKSAWBL	Same as DKSAWB but only incremented for anticipatory writes from below the bar Offset: Table A-7 on page 572.
DKSAWW	Anticipatory writes from the anticipatory write windows set by LDKBMWND and/or LDKBMWNG Offset: Table A-7 on page 572.
DKSAWWL	Same as DKSAWW but only incremented for anticipatory writes from below the bar
DKSDIR	High-water mark of modified (dirty) buffers found within the DKBM statistics window which is created by the LDKBMW parameter. This parameter is not recommended due to the extremely heavy performance penalty imposed by a non-zero value. Offset: see Table A-7 on page 572. Parameter setting required: see Table 13-3 on page 303.
DKSDIRT	The total of all modified buffers found within the DKBM statistics window Offset: see Table A-7 on page 572. Parameter setting required: see Table 13-3 on page 303.
DKSFBS	Times a buffer was needed, and a scan was necessary, because the oldest free buffer was not immediately available Offset: see Table A-7 on page 572.
DKSKIP	The highest number of buffers skipped during any search for a free buffer. A buffer is skipped because it is not immediately available. Offset: see Table A-7 on page 572.
DKSKIPT	The total of all buffers skipped when looking for a free buffer Offset: see Table A-7 on page 572.
DKSRR	Number of times a page that was expected to be in the buffer pool could not be located, which necessitates a physical I/O to disk Offset: see Table A-7 on page 572.
DKSRRFND	Number of times that a page that was expected to be in the buffer pool was located there, which eliminates the need for a physical I/O
DKSTBLA	Table A reads causing real I/O Offset: see Table A-7 on page 572.

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
DKSTBLB	Table B reads causing real I/O Offset: see Table A-7 on page 572.
DKSTBLC	Table C reads causing real I/O Offset: see Table A-7 on page 572.
DKSTBLD	Table D reads causing real I/O Offset: see Table A-7 on page 572.
DSKTBLE	Table E reads causing real I/O Offset: see Table A-7 on page 572.
DKSTBLF	FCT reads causing real I/O Offset: see Table A-7 on page 572.
DKSTKQC	Times the current Table B page had to be closed to allow a fourth buffer to be opened Offset: see Table A-7 on page 572.
DKSWRP	The high water mark of buffers with writes outstanding within the DKBM statistics window, created by the parameter LDKBMW. Rocket Software advises against setting this parameter to a non-zero value due to the extreme performance penalty imposed. Offset: see Table A-7 on page 572. Parameter setting required: see Table 13-3 on page 303.
DKSWRPT	The total number of buffers with outstanding writes within the DKBM statistics window Offset: see Table A-7 on page 572. Parameter setting required: see Table 13-3 on page 303.
DKUPTIME	Milliseconds of DKUPDT time
DKWR	Number of physical page writes to Model 204 files
DKWRL	Number of physical page writes from the below the bar pool
DUMP	Writes to Model 204 file backups (DUMP command output)
DUPDTS	Records written to a deferred update data set for the file
ECCALL	Number of External Call Facility calls
ECCNCT	External Call Facility - Elapsed time for external program
ECCTOUT	Number of External Call Facility calls that timed out (program or subtask unavailable)
ECCWAITM	ECF - Number of calls that waited for module to become available

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
ECCWAITS	Number of External Call Facility calls that waited for a subtask to be come available
ECDELETE	Number of EXTERNAL DELETE calls to External Call Facility
ECLOAD	Number of EXTERNAL LOAD calls to External Call Facility
ECMODMAX	ECF - High-water mark of modules loaded
ECNAMMAX	External Call Facility - High-water mark of call names defined
ECTSKMAX	External Call Facility - High-water mark of subtasks active
ECTWAITM	External Call Facility - Elapsed time spent waiting for a module to become available
ECTWAITS	External Call Facility - Elapsed time spent waiting for a subtask to become available
ERRPDL	High-water mark of the system pushdown list
FBWT	Number of waits for a disk buffer
FINDS	Number of FIND statements evaluated
FSCB	High-water mark of bytes used in the FSCB
FSCBSW	Number of full-screen buffer swaps
FTBL	High-water mark of FTBL appearing in CMPL and EVAL lines for requests containing a field name reference in group context. If one or more groups are opened but no field name references are made, the portion of FTBL used by the open groups is not reported.
GTBL	High-water mark of GTBL
GTBLRS	Number of GTBL rearrangements required to add a global object
GTBLRU	Number of GTBL rearrangements required to add a global string variable
HEAP	Dynamic memory high-water mark
IN	Number of terminal input lines
INCMFS	Input lines from CMS full screen
INCMIO	Input lines from CMS non-full screen
INCRAM	Input lines from Remote User Language threads (IODEV 29)
INVMFS	Input lines from IUCV or VMCF full-screen thread

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
INVMIF	Input arguments supplied by certain IFAM2 or IFAM4 calls from CMS programs
INVMIO	Input lines from IUCV or VMCF non-full screen thread
INXX	Input lines from QSAM users (IODEV 3)
ITBL	High-water mark of ITBL
IXADD	Number of index entries added to Tables C and D, including attempts to add duplicates
IXDEL	Number of entries deleted from Tables C and D
LKPOST	Times an MP subtask posted another task to indicate that an MP lock had become available
LKWAIT	Times MP subtasks invoked z/OS WAIT macros to wait for an available MP lock
LONGUPDTIME(MS)	Total milliseconds of 'too long' update units Parameter setting required: see Table 13-3 on page 303.
LONGUPDTS	Number of 'too long' update units Parameter setting required: see Table 13-3 on page 303.
LWTIM	Elapsed time (ms), spent waiting for a multiprocessing lock. Reported for each MP and zIIP subtask at Model 204 termination or in MONITOR TASKS output (only on Model 5 terminals).
MAXIOX	Number of times Model 204 waited for a free database buffer
MOVE	Number of times the scheduler switched from one user to another
MPLKPREM	Total elapsed time in milliseconds, across the maintask and all subtasks, the Online spent waiting due to operating system preemption. This is the elapsed time between when an MP lock becomes available (lock post) making a task ready to run, and when the task actually gets the CPU. That preemption delay is caused by the operating system dispatching other tasks ahead of this task. Only collected in MP/204 systems.
MPLKWTIM	Total elapsed time in milliseconds, across the maintask and all subtasks, the Online spent waiting for MP locks Only collected in MP/204 systems.
MQAPICNT	MQ/204 - Count of MQ/Series API calls
MQAPITIM	MQ/204 - Elapsed time of MQSeries API calls

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
MQBYTEIN	MQ/204 - Total bytes retrieved (MQGET)
MQBYTEOU	MQ/204 - Total bytes sent (MQPUT/MQPUT1)
MQGETS	MQ/204 - Number of MQGETs performed
MQGWTCNT	MQ/204 - Number of MQGETs with nonzero, non-unlimited wait
MQGWTSUC	MQ/204 - Number of MQGETs with nonzero, non-unlimited wait that succeeded
MQGWTTIM	MQ/204 - Elapsed time for MQGETs with nonzero, non-unlimited wait
MQGWTTSP	MQ/204 - Total wait time specified on MQGETs with nonzero, non-unlimited wait
MQHWQU	MQ/204 - High-water mark of queues concurrently in use
MQHWTASK	MQ/204 - High-water mark of MQ/204 subtasks in use
MQNUMQM	MQ/204 - Number of distinct queue managers connected
MQNUMQU	MQ/204 - Number of distinct queues accessed
MQPUTS	MQ/204 - Number of MQPUTs performed
MQWTM	Mean queue wait time, the average time a subtask spent waiting for work (waiting for PCBs to come into the offload queue)
MTDEQ	The number of times that the MAINTASK took a unit of work from the MAINTASK queue and processed it
MTSDEQ	The number of times that the MAINTASK took a unit of work from an MP subtask queue and processed it because the MAINTASK was idle
MTZDEQ	The number of times that the MAINTASK took a unit of work from a zIIP subtask queue and processed it because the MAINTASK was idle
NTBL	High-water mark of NTBL
OFFIN	Count of records read in by the OFFLOAD subtask
OFFOU	Count of records written to the OFFLOAD stream
OUT	Number of terminal output lines
OUTCMFS	Output lines to CMS full-screen thread
OUTCMIO	Output lines to CMS line-by-line thread
OUTCRAM	Output lines from remote User Language threads (IODEV 29)

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
OUTPB	High-water mark of OUTPB
OUTVMFS	Output lines to IUCV or VMCF full-screen thread
OUTVMIF	Output lines to IFAM2 or IFAM4 calls from CMS programs
OUTVMIO	Output lines to IUCV or VMCF line-by-line threads
OUTXX	Output lines to QSAM users (IODEV 3)
PBRFLT	Number of private buffer reservation faults
PCPU	Percentage of time Model 204 was given the CPU when it wanted CPU time. In an MP environment, system PCPU is the sum of CPU time in all tasks divided by the sum of elapsed time in all tasks.
PDL	High-water mark of the pushdown list
PETIM	Elapsed time (ms), spent waiting due to OS preemption. Reported for the MAINTASK and each MP and zIIP subtask at Model 204 termination or in MONITOR TASKS output (only on Model 5 terminals)
PNDGTIME	Milliseconds of unnecessary broken time
PR	Real time consumed, in milliseconds. $PCPU = CPU / PR$.
QTBL	High-water mark of QTBL
RECADD	Number of records started in Table B, not including extension records
RECDL	Number of records deleted from Table B, not including extension records or records deleted by DELETE RECORDS or IFDSET
RECDS	Number of records processed by FOR statements, SORT statements, IFGET calls, or IFPOINT calls, or similar SQL record processing
REDY	Percentage of performance samples at which the user was ready to run (reported only if the performance subtask is active) Parameter setting required: see Table 13-3 on page 303.
REQ	Number of User Language requests evaluated
REST	Number of reads from a Model 204 file backup (RESTORE command)
RETRYA	Page retries in Table A
RETRYC	Page retries in Table C

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
RQTM	Elapsed time for the activity being reported, exclusive of terminal I/O time
RSXCOMP	Number of compactions of the record locking table. If greater than 0, increase LRETBL.
RUNG	Percentage of performance samples at which the user was running (reported only if the performance subtask is active) Parameter setting required: see Table 13-3 on page 303.
SCHDCPU	In MP configuration, total scheduler CPU for the user or request
SCREENS	Total number of full-screen reads (READ SCREEN statements) evaluated by a User Language request.
SGMTI	Number of input lines from included procedures
SGMTO	Number of output lines to procedures defined by the user
SLIC	Number of times that the user was time-sliced by the Model 204 scheduler
SMPLS	Number of performance sample points at which the user was active or total number of performance samples taken at the system level (reported only if the performance subtask is active) Parameter setting required: see Table 13-3 on page 303.
SORTS	Number of User Language SORT statements or SQL sorts evaluated
SQLI	Remote SQL input high-water mark
SQLO	Remote SQL output high-water mark
SQRD	Number of terminal input lines
SQWR	Number of terminal output lines
STBL	High-water mark of STBL
STCPU	Total amount of time, in milliseconds, that the user has run in an MP offload subtask
STDEQ	Number of times that an MP subtask took a unit of work from an MP subtask queue and processed it
STIMERS	Total number of STIMER or STIMERM calls
STPOST	Times MP subtasks performed real operating system POSTs to send work to the maintask
STRECDs	Number of records processed by SORT statements or SQL sorts

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
STWAIT	Number of times MP subtasks invoked z/OS WAIT macros to wait for work from the maintask
STZDEQ	Number of times that an MP subtask took a unit of work from a zIIP subtask queue and processed it because the MP subtask was idle
SVAC	Active servers (an average, included on a performance line)
SVMX	High-water mark for servers (0 if no server swapping)
SVPAGES	SVPAGES * 4096 is the number of bytes transferred as a result of server reads and writes
SVRD	Number of server reads
SVWR	Number of server writes
SWPG	Percentage of performance samples at which the user was swapping in or out (reported only if the performance subtask is active)
SWT	Times Model 204 issued a real WAIT while not in user-switching mode
TEMX	High-water mark of CCATEMP pages used in the expansion area
TFMX	High-water mark of CCATEMP pages used
TSMX	High-water mark of CCATEMP pages used in the small model page pool
TTBL	High-water mark of TTBL
UBUFHWS	High-water mark, in bytes, of user's Universal Buffer
UDD	Number of lines written to a directed output (USE) data set
UPDETIME(MS)	Number of milliseconds that a file was being updated UPT(MS) appears as the column header for this statistic on a report
UPDETIME	Number of milliseconds that the user was actively updating at least one file
USMX	High-water mark for simultaneous active users (reported only if the performance subtask is active)
USRS	Average number of active users (reported only if the performance subtask is active) Parameter setting required: see Table 13-3 on page 303.
VTBL	Number of VTBL entries at the end of compilation (CMPL) processing or the end of evaluation (EVAL) processing

Table A-1. Statistics with descriptions(Continued)

Statistic	Records...
WAIT	Number of real operating system waits
WTCFR	Number of times that the user waited to obtain a critical file resource, either share or exclusive Parameter setting required: see Table 13-3 on page 303.
WTRLK	Number of times that the user waited to obtain a record lock Parameter setting required: see Table 13-3 on page 303.
WTSV	Percentage of performance samples at which the user was waiting for a server (reported only if the performance subtask is active) Parameter setting required: see Table 13-3 on page 303.
XTBL	High-water mark of XTBL
ZCPU	The CPU (ms) consumed on a zIIP processor by the MAINTASK, an MP subtask or a zIIP subtask. Reported for each task at Model 204 termination and in MONITOR TASKS output (for Model 5 terminals only)
ZTDEQ	The number of times that a zIIP subtask took a unit of work from a zIIP subtask queue and processed it

Header and trailer entries (Type 0)

Each journal block begins with a header record and ends with a trailer record.

Header entries (Type 0)

Header entries have the format shown in Table A-2.

Table A-2. Header entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entire journal record, including header and trailer
2(2)	1	Binary	Type indicator = X'00'
3(3)	1	Binary	Flags used by recovery
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH
12(0C)	4	Binary	Sequence number

Table A-2. Header entry formats(Continued)

Offset dec(hex)	Length	Data type	Description
16(10)	10	Binary	System Store Clock Extended (STCKE) value to provide picosecond accuracy on the journal date/time stamp Bytes 1-9 byte 16
26(1A)	2	Binary	Length of header, type 00
28(1C)	2	Binary	Version in hex
30(1E)	8	EBCDIC	Version in EBCDIC
38(26)	8	EBCDIC	Job name
46(2E)	8	EBCDIC	JES Job ID
54(36)	4	EBCDIC	SMF System ID

- If the flag byte at offset 3 of type 00 has the X'02' bit on, then it is a header record from Model 204 release 7.4.0 or later with its length at offset X'1A'.
- If that flag byte has the X'01' bit on, then the header is from version 7.1.0 and is 26 bytes in length.
- If neither of those bits is on, then the header is from a pre-7.1.0 version and is 16 bytes in length.

Recovery flags

Flags used by recovery can have the value of X'00' or a sum of any of the following:

Value	Block contains these records...
X'80'	Type 1
X'40'	Type 2
X'20'	Type 3
X'10'	Type 4
X'08'	Type 5
X'04'	Type 6

Sequence numbers

The sequence number represents the number of each journal record, starting at 0 and increasing by 1. A merged journal (output of the MERGEJ utility) sequence number consists of a 2-byte hexadecimal number at offset 12(0C),

which indicates the input file this journal record came from, and a 2-byte hexadecimal field at offset 14(0E), which indicates the sequence within the INPUT journal.

Trailer entries (Type 0)

Trailer entries have the format shown in Table A-3.

Table A-3. Trailer entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry = X'000C'
2(2)	1	Binary	Type indicator = X'00'
3(3)	1	—	Unused
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH

Recovery entries (Types 1–6)

Entry types X'01' through X'06' contain information used for recovery. They follow the same general format as all other journal entries. There is *no* type X'07' journal entry.

The format of the recovery entries is meaningful only in the context of the Model 204 ROLL FORWARD algorithm. However, entry types might be of interest in some statistics gathering applications, as follows:

Type	Contains recovery information from...
1	Discontinuity
2	Begin update unit. More than one type 2 entry can be written for each update unit. The update unit number appears as a 4-byte binary number at offset 16(10).
3	End update unit. The update unit number appears in the same format and location as the type 2 entry.
4	File state information: opened, closed, dumped, file parameter list (FPL) page preimage logged or open at the time of a checkpoint
5	Checkpoint
6	File update

System statistics entries (Type 8)

Cumulative statistics, listed by offset in Table A-6 on page 571, are kept for the entire system. Final, partial, and performance system statistics are written to the journal during Model 204 termination in a line that begins with:

```
ST $$$ SYSTEM='version#'
```

For additional information about:

- Using system statistics for performance monitoring, see Chapter 18: “Performance Monitoring and Tuning”.
- Disk buffer monitor statistics and parameters, see “Disk buffer monitor statistics and parameters” on page 435.

Final statistics

The system final statistics line, similar to the following, is produced as part of system termination:

```
ST $$$ SYSTEM='5A 'AUDIT=43 OUT=38 IN=14 WAIT=19 ...
```

Partial statistics

Statistics in a system partial statistics line, similar to the following, accumulate from the beginning of the run, if the User 0 ACCTIM parameter is nonzero:

```
ST $$$ SYSTEM='6.1.0'PARTIAL=3 AUDIT=19 OUT=10 IN=5
WAIT=3 ...
```

Performance statistics

System performance statistic lines, similar to the following, are written if the User 0 RPTCNT and SMPLTIM parameters are nonzero:

```
ST $$$ SYSTEM='6.1.0' PERFORMANCE=n
```

The numbers determined at the performance sample points are averaged for the entire run.

Entry formats

System statistics entries have the format shown in Table A-4.

Table A-4. System statistics entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry

Table A-4. System statistics entry formats (Continued)

Offset dec(hex)	Length	Data type	Description
2(2)	1	Binary	Type indicator = X'08'
3(3)	1	Binary	Subtype indicator: X'00' = final statistics X'01' = partial statistics X'02' = performance statistics X'04' = additional disk buffer monitor X'08' = subtask statistics
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH
12(0C)	2	Binary	Subtask number if subtype = X'08'; otherwise unused
14(0E)	2	Binary	Unused
16(10) to end of journal	—	—	Contents depend on subtypes, shown in tables under the following headings: <ul style="list-style-type: none"> • “System final and partial statistics” on page 564 • “System performance statistics” on page 571 • “Additional disk buffer monitor statistics” on page 572 • “User since-last statistics” on page 579

System final and partial statistics

For subtypes X'00', final statistics, and X'01', partial statistics, the layout of the remainder of the entry is shown in Table A-5. For a description of each statistic see Table A-1 on page 548. The following statistics are Data type Binary. However, the length varies—either 4 or 8, to accommodate 64-bit system architecture.

Table A-5. System final and partial statistics

Offset dec(hex)	Length	Description
16(10)	8	AUDIT
24(18)	8	OUT

Table A-5. System final and partial statistics (Continued)

Offset dec(hex)	Length	Description
32(20)	8	IN
40(28)	8	OUTXX
48(30)	8	INXX
56(38)	8	DEV5
64(40)	8	DEV6
72(48)	8	DEV7
80(50)	8	DEV8
88(58)	8	DEV9
96(60)	8	DEV10
104(68)	8	DEV11
112(70)	8	DEV12
120(78)	8	DEV13
128(80)	8	DEV14
136(88)	8	OUTTTY
144(90)	8	INTTY
152(98)	8	DEV17
160(A0)	8	DEV18
168(A8)	8	DEV19
176(B0)	8	DEV20
184(B8)	8	Spare
192(C0)	8	Spare
200(C8)	8	DEV23
208(D0)	8	DEV24
216(D8)	8	DEV25
224(E0)	8	DEV26
232(E8)	8	DEV27
240(F0)	8	DEV28
248(F8)	8	OUTCRAM
256(100)	8	INCRAM
264(108)	8	DEV31

Table A-5. System final and partial statistics (Continued)

Offset dec(hex)	Length	Description
272(110)	8	DEV32
280(118)	8	Spare
288(120)	8	Spare
296(128)	8	DEV35
304(130)	8	DEV36
312(138)	8	DEV37
320(140)	8	DEV38
328(148)	8	OUTVMIO
336(150)	8	INVMIO
344(158)	8	OUTVMFS
352(160)	8	INVMFS
360(168)	8	OUTVMIF
368(170)	8	INVMIF
376(178)	8	OUTCMIO
384(180)	8	INCMIO
392(188)	8	OUTCMFS
400(190)	8	INCMFS
408(198)	8	DEV49
416(1A0)	8	DEV50
424(1A8)	8	OFFIN
432(1B0)	8	OFFOU
440(1B8)	8	DEV53
448(1C0)	8	DEV54
456(1C8)	8	DEV55
464(1D0)	8	DEV56
472(1D8)	8	DEV57
480(1E0)	8	DEV58
488(1E8)	8	DEV59
496(1F0)	8	DEV60
504(1F8)	8	DEV61

Table A-5. System final and partial statistics (Continued)

Offset dec(hex)	Length	Description
512(200)	8	DEV62
520(208)	8	DEV63
528(210)	8	DEV64
536(218)	8	DEV65
544(220)	8	DEV66
552(228)	8	DEV67
560(230)	8	DEV68
568(238)	8	DEV69
576(240)	8	DEV70
584(248)	8	DEV71
592(250)	8	DEV72
600(258)	8	DEV73
608(260)	8	DEV74
616(268)	4	WAIT
620(26C)	4	MPLKWTIM
624(270)	8	DKRD
632(278)	8	DKWR
640(280)	4	SVRD
644(284)	4	SVWR
648(288)	8	CPU
656(290)	8	REQ
664(298)	8	MOVE
672(2A0)	4	DUMP
676(2A4)	4	REST
680(2A8)	4	SLIC
684(2AC)	4	CNCT
688(2B0)	4	FBWT
692(2B4)	4	SWT
696(2B8)	4	ERRPDL
700(2BC)	4	MPLKPREM

Table A-5. System final and partial statistics (Continued)

Offset dec(hex)	Length	Description
704(2C0)	8	RECADD
712(2C8)	8	RECDEL
720(2D0)	8	BADD
728(2D8)	8	BDEL
736(2E0)	8	BCHG
744(2E8)	8	IXADD
752(2F0)	8	IXDEL
760(2F8)	8	FINDS
768(300)	4	SORTS
772(304)	4	Spare
776(308)	8	RECDS
784(310)	8	STRECDS
792(318)	8	DKAR
800(320)	8	DKPR
808(328)	4	DKRR
816(330)	4	TFMX
820(334)	4	USMX
824(338)	4	SVMX
828(33C)	4	Spare
832(340)	8	APSYLD
840(348)	8	APSYLDD
848(350)	4	APSYLDT
852(354)	4	Spare
856(358)	8	DKPRF
864(360)	4	SMPLS
868(364)	4	USRS
872(368)	4	SVAC
876(36C)	4	RUNG
880(370)	4	REDY
884(374)	4	BLKI

Table A-5. System final and partial statistics (Continued)

Offset dec(hex)	Length	Description
888(378)	4	WTSV
892(37C)	4	BLKO
896(380)	4	SWPG
900(384)	4	PCPU
904(388)	4	DIRRCD
908(38C)	4	Spare
912(390)	8	STCPU
920(398)	8	STDEQ
928(3A0)	8	STWAIT
936(3A8)	4	STPOST
940(3AC)	4	LKWAIT
944(3B0)	4	LKPOST
948(3B4)	4	RSXCOMP
952(3B8)	4	SCHDCPU
956(3BC)	4	SCREENS
960(3C0)	4	STIMERS
964(3C4)	4	Spare
968(3C8)	8	SVPAGES
976(3D0)	4	PBRSLT
980(3D4)	4	COMMITTS
984(3D8)	4	BACKOUTS
988(3DC)	4	LONGUPDTS
992(3E0)	4	LONGUPDTIME(MS)
996(3E4)	4	Spare
1000(3E8)	4	CDLWAIT
1004(3EC)	4	MQGETS
1008(3F0)	4	MQPUTS
1012(3F4)	4	UBUFHWS
1016(3F8)	4	MQHWTASK
1020(3FC)	4	Spare

Table A-5. System final and partial statistics (Continued)

Offset dec(hex)	Length	Description
1024(400)	8	MQBYTEIN
1032(408)	8	MQBYTEOU
1040(410)	4	MQHWQU
1044(414)	4	MQNUMQU
1048(418)	4	MQNUMQM
1052(41C)	4	Spare
1056(420)	8	MQAPITIM
1064(428)	8	MQAPICNT
1072(430)	8	MQGWTTIM
1080(438)	4	MQGWTCNT
1084(43C)	4	Spare
1088(440)	8	MQGWTTSP
1096(448)	4	MQGWTSUC
1100(44C)	4	ECLOAD
1104(450)	4	ECDELETE
1108(454)	4	Spare
1112(458)	8	ECCALL
1120(460)	4	ECCWAITM
1124(464)	4	ECCWAITS
1128(468)	4	ECTWAITM
1132(46C)	4	ECTWAITS
1136(470)	4	ECCTOUT
1140(474)	4	ECCNCT
1144(478)	4	ECMODMAX
1148(47C)	4	ECNAMMAX
1152(480)	4	ECTSKMAX
1156(484)	4	GTBLRU
1160(488)	4	GTBLRS
1164(48C)	4	TSMX
1168(490)	4	TEMX

Table A-5. System final and partial statistics (Continued)

Offset dec(hex)	Length	Description
1172(494)	4	MAXIOX
1176(498)	8	DKRDL
1184(4A0)	8	DKWRL
1192(4A8)	8	CPUTOTZE
1200(4B0)	8	CPUONZIP
1208(4B8)	8	Spare
1216(4C0)	8	Spare
1224(4C8)	8	Spare
1232(4D0)	8	Spare
1240(4D8)	8	Spare
1248(4E0)	8	MTDEQ
1256(4E8)	8	MTSDEQ
1264(4F0)	8	MTZDEQ
1272(4F8)	8	STZDEQ
1280(500)	8	ZTDEQ

System performance statistics

Entry format

For subtype X'02', system performance statistics, the layout of the remainder of the entry is shown in Table A-6. Each statistic in the table is: Length, 4; Data type, Binary. See Table A-1 on page 548 for the meaning of each statistic.

Table A-6. Layout of system subtype X '02' entries

Offset dec(hex)	Statistic
16(10)	SMPLS
20(14)	USRS
24(18)	SVAC
28(1C)	RUNG
32(20)	REDY
36(24)	BLKI
40(28)	WTSV

Table A-6. Layout of system subtype X '02' entries

Offset dec(hex)	Statistic
44(2C)	BLKO
48(30)	SWPG

Additional disk buffer monitor statistics

For subtype X'04', additional disk buffer monitor statistics, the layout of the entry is shown in Table A-7. Each statistic in the table is: Length, 8; Data type, Binary. For a description of each statistic, see Table A-1 on page 548.

Table A-7. Disk buffer monitor statistics

Offset dec(hex)	Disk buffer monitor statistic
16(10)	Spare
24(18)	DKSTKQC
32(20)	DKSWRP
40(28)	DKSWRPT
48(30)	DKSDIR
56(38)	DKSDIRT
64(40)	DKSKIP
72(48)	Spare
80(50)	Spare
88(58)	DKSRR
96(60)	DKSFBS
104(68)	DKSKIPT
112(70)	DKSAWT
120(78)	DKSAWW
128(80)	DKSAWWL
136(88)	DKSAWB
144(90)	DKSAWBL
152(98)	DKSRRFND
160(A0)	DKSTBLF
168(A8)	DKSTBLA
176(B0) 8	DKSTBLB
184(B8)	DKSTBLC

Table A-7. Disk buffer monitor statistics (Continued)

Offset dec(hex)	Disk buffer monitor statistic
192(C0)	DKSTBLD
200(C8)	Spare
208(D0)	Spare
216(D8)	Spare
224(E0)	Spare
232(E8)	Spare
240(F0)	Spare
248(F8)	Spare
256(100)	DKSTBLX
264(108)	DKSTBLE
272(110)	Spare

Multiprocessing (MP) subtask statistics

The following statistics apply only to installations configured for multiprocessing using MP/204. For subtype X'08', subtask specific statistics, the layout of the entry is shown in Table A-8. For the meaning of each statistic, see Table A-1 on page 548.

Table A-8. MP subtask statistics and offsets

MP statistic	Offset dec(hex)	Length
CPU	16(10)	8
PR	24(18)	8
PCPU	32(20)	4
STWAIT	36(24)	4
STPOST	40(28)	4
LKWAIT	44(2C)	4
LKPOST	48(30)	4
SPARE	52(34)	4
MQWTM	56(38)	8
DEQ	64(40)	8
ZTDEQ	72(48)	8
STDEQ	80(50)	8

Table A-8. MP subtask statistics and offsets

MP statistic	Offset dec(hex)	Length
LWTIM	88(58)	4
PETIM	92(5C)	4
ZCPU	96(60)	8

User statistics entries (Type 9)

User lines report on an individual user's activity. These statistics include:

- Final (logout) and partial statistics
- Since-last and ordered index statistics
- Performance statistics

Each user line starts with the following statistics:

```
ST $$$ USERID='userid' ACCOUNT='account'
```

The subtype, in this case PERFORMANCE, follows USERID and ACCOUNT statistics:

```
ST $$$ USERID='userid' ACCOUNT='account'
PERFORMANCE=n
```

The format of user statistics entries is shown in Table A-9.

Table A-9. User statistics entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry
2(2)	1	Binary	Type indicator = X'09'
3(3)	1	Binary	Subtype indicator: <ul style="list-style-type: none"> • X'00' = logout statistics • X'01' = since-last statistics • X'02' = partial statistics • X'04' = performance statistics • X'81' = since-last statistics including conflict statistics
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH
12(0C)	2	Binary	Server number

Table A-9. User statistics entry formats

Offset dec(hex)	Length	Data type	Description
14(0E)	2	Binary	User number
16(10)	10(0A)	EBCDIC	Account: <ul style="list-style-type: none"> • 2nd argument of LOGON/LOGIN command • Default is user ID
26(1A)	10(0A)	EBCDIC	User ID
36(24)	4	EBCDIC	Since-last ID = See Table A-11 on page 580 where the possible activity types are listed.
40(28) to end of journal		Binary	Contents depend on subtypes, shown in: <ul style="list-style-type: none"> • Table A-10 on page 576 • Table A-12 on page 581 • Table A-13 on page 584 • Table A-14 on page 585 • Table A-15 on page 586

User final and partial statistics

Final (logout) statistics

Final (logout) statistics, listed in Table A-10 on page 576, summarize a user's activities since the last login. A LOGOUT statistics line, such as the following, is written after each logout if the parameter following ACCOUNT is not PARTIAL, LAST, or PERFORMANCE:

```
yydddhhmmssnnnsssuuuuu ST $$$ ACCOUNT='NO ACCOUNT' SQRD=5
...
```

- *yyddd* is the year and Julian day.
- *hhmmss* is the time of day in hours, minutes, and seconds.
- *nnn* is a counter to distinguish lines produced in the same second.
- *sss* is the number of the server currently handling the user. Leading zeros are suppressed.
- *uuuuu* is a 5-digit user number associated with the audit trail line. Leading zeros are suppressed.

In some cases, the account name appears as “NO USERID NO ACCOUNT”. Statistics in this category are not chargeable to any one user and should be considered system overhead.

Final lines are recognized by the absence of an indicator for another type of line.

Partial statistics

Optional partial statistics, listed in Table A-10 on page 576, can be provided for a run up to a few minutes before a system failure that prevents normal termination and production of final user and system statistics.

User partial statistics lines, such as the following, are made available for active users:

```
ST $$$ ACCOUNT='NO ACCOUNT' PARTIAL=21 ...
```

Statistics accumulate either:

- As long as the user is logged in
- or
- From the first terminal input to disconnection when a login is not required

Partial statistics are requested by setting the time interval, in minutes, between successive partial lines on User 0's parameter line with the ACCTIM parameter. ACCTIM must be nonzero.

The contents of the statistics portion of the entry, starting at offset 40(28), depends upon the subtype of the entry.

Partial statistics lines are indicated on the journal by the parameter PARTIAL=*n*, where *n* is a counter from the beginning of the run. PARTIAL=*n* immediately follows the type parameter.

For subtypes X'00', logout statistics, and X'02', partial statistics, the layout of the remainder of the entry is shown in Table A-10. Each statistic in the table is: Length, 4; Data type, Binary. For the meaning of each statistic, see Table A-1 on page 548.

Table A-10. Layout of user subtype X '00' and X '02' entries

Offset dec(hex)	Statistic
40(28)	CNCT
44(2C)	DKRD
48(30)	DKWR
52(34)	SQRD
56(38)	SQWR
60(3C)	SGMTI
64(40)	SGMTO

Table A-10. Layout of user subtype X '00' and X '02' entries (Continued)

Offset dec(hex)	Statistic
68(44)	SVRD
72(48)	SVWR
76(4C)	CPU
80(50)	REQ
84(54)	MOVE
88(58)	DUMP
92(5C)	REST
96(60)	SLIC
100(64)	AUDIT
104(68)	WAIT
108(6C)	FBWT
112(70)	UDD
116(74)	RECADD
120(78)	RECDEL
124(7C)	BADD
128(80)	BDEL
132(84)	BCHG
136(88)	IXADD
140(8C)	IXDEL
144(90)	FINDS
148(94)	SORTS
152(98)	RECDS
156(9C)	STRECDS
160(A0)	DKAR
164(A4)	DKPR
168(A8)	DKRR
172(AC)	COMMITTS
176(B0)	BACKOUTS
180(B4)	UPDTTIME(MS)
184(B8)	LONGUPDTS

Table A-10. Layout of user subtype X '00' and X '02' entries (Continued)

Offset dec(hex)	Statistic
188(BC)	LONGUPDTIME(MS)
192(C0)	SMPLS
196(C4)	RUNG
200(C8)	REDY
204(CC)	BLKI
208(D0)	WTSV
212(D4)	BLKO
216(D8)	SWPG
220(DC)	PCPU
224(E0)	DIRRCD
228(E4)	BXCHNG
232(E8)	BXDELE
236(EC)	BXNEXT
240(F0)	BXFIND
244(F4)	BXINSE
248(F8)	BXSPLI
252(FC)	BXRFND
256(100)	BXFREE
260(104)	STCPU
264(108)	STDEQ
268(10C)	SCHDCPU
272(110)	SCREENS
276(114)	SVPAGES
280(118)	PBRSFLT
284(11C)	MQGETS
288(120)	MQPUTS
292(124)	UBUFHWS
296(128)	MQHWTASK
300(12C)	MQBYTEIN
304(130)	MQBYTOU

Table A-10. Layout of user subtype X '00' and X '02' entries (Continued)

Offset dec(hex)	Statistic
308((134)	MQHWQU
312(138)	MQNUMQU
316(13C)	MQNUMQM
320(140)	MQAPITIM
324(144)	MQAPICNT
328(148)	MQGWTTIM
332(14C)	MQGWTCNT
336(150)	MQGWTTSP
340(154)	MQGWTSUC
344(158)	ECLOAD
348(15C)	ECDELETE
352(160)	ECCALL
356(164)	ECCWAITM
360(168)	ECCWAITS
364(16C)	ECTWAITM
368(170)	ECTWAITS
372(174)	ECCTOUT
376(178)	ECCNCT
380(17C)	GTBLRU
384(180)	GTBLRS
388(184)	FSCBSW
392(188)	Spare

User since-last statistics

Since-last statistics

Statistics are kept for each User Language request, host language interface call, and a number of Model 204 system commands. When each request or command completes, a since-last statistics line is written. The line includes the ACCOUNT parameter at the beginning, followed by the LAST parameter:

Several MONITOR commands produce displays containing the user's last or current activity in the FUNC column. When this activity completes, statistics are

also written to CCAAUDIT and the activity is described in ST lines as LAST='activity'.

Syntax ST \$\$\$ USERID='userid' ACCOUNT='accountname' LAST='acty'

Where acty can be one of the following.

Table A-11. Activity types

Option	Meaning
BLDR	BLDREUSE command
BLDX	Z command
CMPL	Compilation of a User Language request
CPTB	COMPACTB command
CPYP	COPY PROCEDURE command
CREA	CREATE FILE <i>filename</i> command
DISP	DISPLAY PROCEDURE command
DUMP	DUMP file command
EDIT	Editing a request
EVAL	Evaluation of a User Language request
EXEC	Execution of an SQL request
FLDC	Compilation of a FLOD procedure
FLDE	Evaluation of a FLOD procedure
IMPC	Implied commit
INCR	IN FILE <i>filename</i> INCREASE DATASETS command
INIT	IN FILE <i>filename</i> INITIALIZE command
LOAD	Loading a precompiled (ASPY) procedure
PREP	Compilation of an SQL request
REST	RESTORE file command

Since-last high-water marks

The since-last statistics include high-water marks of various work tables used by User Language requests. Values are reported in terms of the units in which each table is allocated. For example, QTBL is allocated in units of 16 bytes.

Since-last in relation to final and partial statistics

Many since-last statistics, such as CNCT, CPU, and DKRD, are also reported on user final and partial statistics lines. (See Table A-1 on page 548.) The statistics on these lines are roughly equal to the sum of all of the preceding since-last statistics for the user. Because since-last statistics are not maintained for every Model 204 command, final statistics are generally larger.

For subtype X'01', since-last statistics, the layout of the remainder of the entry begins as shown in Table A-12. For the meaning of each statistic in Table A-12 and Table A-13, see Table A-1 on page 548. The layout of the end of the entry depends on whether you chose to collect conflict statistics:

- If you do not collect conflict statistics, the end of the entry is described in Table A-13.
- If you do collect conflict statistics, the end of the entry is described in Table A-14. When you collect conflict statistics, the subtype changes to X'81'.

Table A-12. Layout of user subtype X '01' and X'81' entries

Offset dec(hex)	Statistic
40(28)	NTBL
44(2C)	GTBL
48(30)	QTBL
52(34)	STBL
56(38)	TTBL
60(3C)	VTBL
64(40)	PDL
68(44)	FTBL
72(48)	XTBL
76(4C)	ITBL
80(50)	FSCB
84(54)	OUTPB
88(58)	HEAP
92(5C)	SQLI
96(60)	SQLO
100(64)	CNCT
104(68)	CPU
108(6C)	DKRD
112(70)	DKWR

Table A-12. Layout of user subtype X '01' and X'81' entries (Continued)

Offset dec(hex)	Statistic
116(74)	UDD
120(78)	OUT
124(7C)	SLIC
128(80)	IN
132(84)	RECADD
136(88)	RECDEL
140(8C)	BADD
144(90)	BDEL
148(94)	BCHG
152(98)	IXADD
156(9C)	IXDEL
160(A0)	FINDS
164(A4)	SORTS
168(A8)	RECDS
172(AC)	STRECDS
176(B0)	PCPU
180(B4)	RQTM
184(B8)	DIRRCD
188(BC)	BXCHNG
192(C0)	BXDELE
196(C4)	BXNEXT
200(C8)	BXFIND
204(CC)	BXINSE
208(D0)	BXSPLI
212(D4)	BXRFND
216(D8)	BXFREE
220(DC)	STCPU
224(E0)	STDEQ
228(E4)	SCHDCPU
232(E8)	SCREENS

Table A-12. Layout of user subtype X '01' and X'81' entries (Continued)

Offset dec(hex)	Statistic
236(EC)	SVRD
240(F0)	SVWR
244(F4)	DKPR
248(F8)	SVPAGES
252(FC)	COMMITTS
256(100)	BACKOUTS
260(104)	UPDTTIME(MS)
264(108)	LONGUPDTS
268(10C)	LONGUPDTIME(MS)
272(110)	MQGETS
276(114)	MQPUTS
280(118)	UBUFHWS
284(11C)	MQHWTASK
288(120)	MQBYTEIN
292(124)	MQBYTEOU
296(128)	MQHWQU
300(12C)	MQNUMQU
304(130)	MQNUMQM
308(134)	MQAPITIM
312(138)	MQAPICNT
316(13C)	MQGWTTIM
320(140)	MQGWTCNT
324(144)	MQGWTTSP
328(148)	MQGWTSUC
332(14C)	ECLOAD
336(150)	ECDELETE
340(154)	ECCALL
344(158)	ECCWAITM
348(15C)	ECCWAITS
352(160)	ECTWAITM

Table A-12. Layout of user subtype X '01' and X'81' entries (Continued)

Offset dec(hex)	Statistic
356(164)	ECTWAITS
360(168)	ECCTOUT
364(16C)	ECCNCT
368(170)	GTBLRU
372(174)	GTBLRS
376(178)	FSCBSW
380(17C)	Spare
384(180)	Spare2
388(184)	Spare3
392(188)	Spare 5
396(18C)	Spare6

Conflict statistics

Whether you are collecting conflict statistics determines the layout at the end of the entry. Two of the conflict statistics keep track of critical file-resource conflicts and two keep track of record-locking conflicts. Whether these statistics are collected and journaled depends on settings of the parameters CFRLOOK and CFRJRNL, described in the *Model 204 Parameter and Command Reference*.

These statistics are not displayed with the output of the MONITOR SL command. Instead, you can view them (with all the other since-last statistics) by issuing the MONITOR command with the keyword CONFLICT.

The remainder of the entry depends on the setting of CFRJRNL.

Table A-13 illustrates the layout when CFRJRNL is set to 0.

Table A-13. Layout of user subtype X '01' entries (CFRJRNL=0)

Offset dec(hex)	Length	Data type	Description
400(190)	10	EBCDIC	Subsystem name
410(19A)	8	EBCDIC	Procedure file name
418(1A2)	1	Binary	Length of following procedure name
419(1A3)	LAUDPROC	EBCDIC	Procedure name

Table A-14 illustrates the layout when CFRJRNL is set to 1. It reflects the change in offset and that the user subtypes are now X'81' entries.

Table A-14. Layout of user subtype X '81' entries (CFRJRNL=1)

Offset dec(hex)	Length	Data type	Entry type
400(190)	4	Binary	BLKCFRE stastic
404(194)	4	Binary	BLKRLK statistic
408(198)	4	Binary	WTCFR statistic
412(19C)	4	Binary	WTRLK statistic
416(1A0)	10	EBCDIC	Subsystem name
426(1AA)	8	EBCDIC	Procedure file name
434(1B2)	1	Binary	Length of following procedure name
435(1B3)	LAUDPROC	EBCDIC	Procedure name

Note: The conflict statistics, BLKCFRE, BLKRLK, WTCFR, and WTRLK, are inexact. Although a user might at times conflict with multiple other users, only the first conflict encountered is counted.

User performance statistics

Optional performance statistics lines provide information about the operation of the Model 204 scheduler during a short period of time. Information about the length of the scheduler queues, the number of active users and servers, and the queue distribution for each user, is obtained by sampling.

A performance line starts with:

```
ST $$$ USERID='userid' ACCOUNT='accountname'
PERFORMANCE=n
```

Parameters to activate performance statistics

To request performance statistics, specify the number of samples to take between performance statistics lines (RPTCNT parameter) and the time interval between samples (SMPLTIM parameter) on User 0's parameter line. When you request scheduler and server performance data (SMPLTIM parameter), an asynchronous operating system task, called the performance subtask, is created. Both RPTCNT and SMPLTIM must be nonzero.

Sampling

The performance subtask spends most of its time in a wait state. Every SMPLTIM milliseconds, it records the current state of the system. Data from the

most recent sample is available at an Online terminal through the MONITOR command, described in “ONLINE monitoring” on page 120.

The number of samples taken during a run (RPTCNT), times the number of users in the run (NUSERS), must be less than 4.2 million. In large systems that run for a very long period of time, the time interval between samples cannot be extremely short. If the limit is exceeded, an OC1 interrupt occurs when the SMPLS statistic value is calculated during the printing of the system statistics, either a performance or final line.

Layout of performance statistics

For subtype X'04', performance statistics, the layout of the remainder of the entry is shown in Table A-15. Each statistic in the table is: Length, 4; Data type, Binary. For the description of each statistic, see Table A-1 on page 548.

Table A-15. Layout of user subtype X '04' entries

Offset dec(hex)	Statistic
40(28)	SMPLS
44(2C)	RUNG
48(30)	REDY
52(34)	BLKI
56(38)	WTSV
60(3C)	BLKO
64(40)	SWPG

File statistics entries (Type 10)

File statistics, summarized in Table A-16, are kept for every file opened during a run, including CCAGRP, CCASYS, and CCATEMP. The activity of a file for all users is summarized from the time the file is first opened until the *last user* issues a CLOSE command. The statistics appear after the last CLOSE command in lines that follow the date and time information (yyyymmddhhmmssnnnnssuuuuu) with:

```
ST $$$ FILE='filename'
```

System termination writes file statistics for each file still open at the end of the run.

Final statistics

A final statistics line similar to the following is produced when the last user of a file closes the file:

```
ST $$$ FILE='PAYABLES' DKRD=25 DKRD=25 DKWR=15 REQ=3 REC-
```

ADD=6 . . .

Partial statistics

File partial statistics lines similar to the following are furnished for each open file:

ST \$\$\$ FILE= 'PAYABLES' PARTIAL=3 . . .

File statistics journal records layout

File statistics entries have the format shown in Table A-16. For a description of each statistic, see Table A-1 on page 548.

Table A-16. File statistics entry formats

Offset dec(hex)	Length	Data type	Type of entry
0(0)	2	Binary	Length of entry
2(2)	1	Binary	Type indicator = X'0A'
3(3)	1	Binary	Subtype indicator: X'00' = file close X'01' = partial
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH
12(0C)	4	—	Unused
16(10)	8	EBCDIC	File name
24(18)	2	—	Unused

Table A-17 lists the file statistics and the *decimal(hexadecimal)* offset. These statistics are Data Type Binary and eight bytes long.

Table A-17. File statistics

File statistic	Offset dec(hex)
DKRD	26(1A)
DKWR	34(22)
REQ	42(2A)
RETRYA	50(32)
RETRYC	58(3A)
DUPDTS	66(42)

Table A-17. File statistics (Continued)

File statistic	Offset dec(hex)
RECADD	74(4A)
RECDEL	82(52)
BADD	90(5A)
BDEL	98(62)
BCHG	106(6A)
IXADD	114(72)
IXDEL	122(7A)
DIRRCD	130(82)
BXCHNG	138(8A)
BXDELE	146(92)
BXNEXT	154(9A)
BXFIN	162(A2)
BXINSE	170(AA)
BXSPLI	178(B2)
BXRFND	186(BA)
BXFREE	194(C2)
UPDTIME	202(CA)
PNDGTIME	210(D2)
DKUPTIME	218(DA)
COMMIT	226(E2)
BACKOUT	234(EA)

Text entries (Type 11, Type 13)

Type 11 (X'0B') and Type 13 (X'0D') entries record most of the actual text of the audit trail.

Noncontinued text entries (Type 11— X'0B')

Each Type 11 journal entry corresponds to one logical entry in the audit trail, even though it might generate more than one line of physical output in the audit trail.

The format of a Type 11 entry is shown in Table A-18.

Table A-18. Noncontinued text entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry
2(2)	1	Binary	Type indicator = X'0B'
3(3)	1	Binary	Subtype indicator: X'01' = LI line X'02' = LP line X'03' = LR line X'04' = CI line X'05' = QT line X'08' = CP line X'10' = OI line X'20' = OO line X'40' = LS line X'80' = CS line
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH
12(0C)	2	Binary	Server number
14(0E)	2	Binary	User number
16(10)	Variable	EBCDIC	Text of audit trail entry

Possibly continued text entries (Type 13 — X'0D')

Each Type 13 journal entry corresponds to one, or part of one, logical entry in the audit trail. See Table 13-4 on page 304 for a definition of the line types.

The format of a Type 13 entry is shown in Table A-19.

Table A-19. Possibly continued text entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry
2(2)	1	Binary	Type indicator = X'0D'

Table A-19. Possibly continued text entry formats

Offset dec(hex)	Length	Data type	Description
3(3)	1	Binary	Subtype indicator: X'01' = US line X'02' = XX line X'04' = RK line X'08' = AD line X'10' = MS line X'20' = ER line
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH
12(0C)	2	Binary	Server number
14(0E)	2	Binary	User number
16(10)	Variable	EBCDIC	Text of audit trail entry

Subtype indicator X'80' summed with any subtype means continuation into the next entry.

Initialization entries (Type 12 — X'0C')

One Type 12 journal entry is written as the first nonheader entry in the journal during initialization. It contains basic information about the run being initialized.

The format of a Type 12 entry is shown in Table A-20.

Table A-20. Initialization entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry = X'001A'
2(2)	1	Binary	Type indicator = X'0C'
3(3)	1	—	Unused
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH
12(0C)	8	EBCDIC	Version ID (for example, 4.2.0G)
20(14)	2	Binary	Spare for the run
22(16)	4	EBCDIC	SMF ID (SYSID parameter)

Time-stamp entries (Type 14 — X'0E')

Type 14 entries are used to force time-stamps to the journal.

The format of a Type 14 entry is shown in Table A-21.

Table A-21. Time-stamp entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry = X'000C'
2(2)	1	Binary	Type indicator = X'0E'
3(3)	1	—	Unused
4(4)	4	Unsigned packed data	Julian date = 0CYYDDDF
8(8)	4	Unsigned packed data	Time = HHMMSSSTH

Merged journal bracketing entries (Type 15)

Type 15 entries appear only in the output of the MERGEJ utility. See “Using the MERGEJ utility” on page 401.

The format of a Type 15 entry is shown in Table A-22.

Table A-22. Merged journal bracketing entry formats

Offset dec(hex)	Length	Data type	Description
0(0)	2	Binary	Length of entry = X '002E'
2(2)	1	Binary	Type indicator = X '0F'
3(3)	1	Binary	Subtype indicator: X'01' = begin bracket X'02' = end bracket
4(4)	12(0C)	—	Unused
16(10)	2(2)	Binary	Number of journals merged together
18(12)	28(1C)	EBCDIC	Text: <ul style="list-style-type: none"> For subtype X'01' = BEGINNING OF MERGED JOURNAL For subtype X'02' = END OF MERGED JOURNAL

System Management Facilities

You can write certain statistics to the System Management Facilities (SMF) data set by:

- Adding an SVC to the operating system

The following message is issued whenever the SMFSVC is not set to a value greater than 199 and less than 256. IBM documentation requires that all user SVCs be in the range 200 - 255 inclusive.

```
M204.2789: SMFSVC = nnn INVALID; SMF ACCOUNTING DIS-  
ABLED
```

- Modifying the program that processes the SMF data set to handle the records written by Model 204

Since-last statistics or logout statistics can be written concurrently with statistics that are written to the journal. Logout statistics are more complete than since-last statistics. Many more since-last records than logout records are usually written.

If the operating system crashes before a user logs out, the SMF data set has a since-last record for every request run, but no logout record.

File and system statistics cannot be written to the SMF data set.

During system initialization, Model 204 issues a message reporting the SMF system ID, the run's job name, the step name, and the JES ID. For operating systems other than z/OS, the JES ID is blank.

z/VM requirements

In the CMS interface to Model 204, you can incorporate a user-written accounting exit that invokes the ACCTEXIT routine to process each statistics record as it is written. The following considerations apply:

- Take care to preserve the CMS environment in which Model 204 is running. Do not change the contents of general registers 13 and 14.
- You must specify the SMFLORN and/or SMFSLRN parameters on User 0's parameter line to activate accounting record production.
- TEXT file for the ACCTEXIT module must be available on an accessed disk when generating M204CMS using M204GEN. For complete installation procedures, see the *Rocket Model 204 z/VM Installation Guide*.

z/OS requirements

To write since-last SMF records in a z/OS environment:

1. Write and install a type 3 or type 2 SVC.

A type 3 or type 2 (for performance enhancement) SVC is required to enable Model 204 to use the privileged macro SMFWTM for writing records to an SMF data set. Write the SVC to issue the SMFWTM macro and return.

Basic assembly language code for this SVC is:

```
SMFWTM    (1)
BR      14
```

2. Add the SVC to the z/OS SVC table or to SYS1.PARMLIB(IEASVCxxx) — whichever is appropriate for the release of z/OS that you are running. If you create a type 2 SVC, you must link it into the z/OS nucleus. If you create a type 3 SVC, install it in LPALIB.

3. Activate the SMF feature.

To activate SMF, set the following parameters on User 0's parameter line:

- SMFSVC — Number of the SMF SVC
- SMFSLRN — SMF record number for since-last records
- SMFLORN — SMF record number for logout records

The following considerations apply:

- SMF SVC must be in an APF-authorized library.
- Since-last SMF records are written only if the SMFSVC and SMFSLRN parameters are set to nonzero values at the beginning of the run.
- Logout SMF records are written only if the SMFSVC and SMFLORN parameters are set to nonzero values at the beginning of the run.
- If all three parameters are set, each type of record is written.

SMF record numbers, which identify the record type to the programs that process the SMF data set, are written as part of the header of the SMF records. The record numbers must be between 128 and 255.

See “System Management Facility record layout and statistics” on page 593.

System Management Facility record layout and statistics

SMF logout record layout

Table A-23 shows the basic format of the Model 204 SMF logout record.

Table A-23. Basic format of Model 204 SMF logout record

Offset(hex)	Length	Description
0(0)	2	Length of record
		(X'0138' if logout record)
		(X'0164' if since last and CFRJRNL=0)
		(X'0174' if since last and CFRJRNL=1)
2(2)	2	Zeros
4(4)	1	System configuration

Table A-23. Basic format of Model 204 SMF logout record (Continued)

Offset(hex)	Length	Description
5(5)	1	Record number (SMFSLRN or SMFLORN)
6(6)	8	Time and date
14(E)	4	System identification
18(12)	8	Job name
26(1A)	8	Time and date of login
34(22)	10	Login user ID
44(2C)	10	Login account
54(36)	2	Binary user number
56(38)	1	X'01' if CFRJRNL=1 (conflict counters present) X'00' if CFRJRNL=0 (conflict counters not present)
57(unused)	3	Unused (reserved)

- Statistics depend upon the record number.
- SMF system identification field (offset 16) is the site-definable 4-character field retrieved from the SMF control area.

SMF control area field contains the CPU identification string that is stored at system IPL time, and can be used to determine on which CPU Model 204 is running in a multi-CPU environment.

Offsets for SMF logout record statistics

Table A-24 on page 594 shows the layout of SMF logout record statistics. Each statistic in the table is: Length, 4; Data type, Binary. This record has the same statistics in the same order as the CCAJRNL logout record, only the offsets are different. For a description of each statistic, see Table A-1 on page 548.

Table A-24. Offset location of SMF logout record statistics

Offset dec(hex)	Statistic
60(3C)	CNCT
64(40)	DKRD
68(44)	DKWR
72(48)	SQRD
76(4C)	SQWR
80(50)	SGMTI
84(54)	SGMTO

Table A-24. Offset location of SMF logout record statistics (Continued)

Offset dec(hex)	Statistic
88(58)	SVRD
92(5C)	SVWR
96(60)	CPU
100(64)	REQ
104(68)	MOVE
108(6C)	DUMP
112(70)	REST
116(74)	SLIC
120(78)	AUDIT
124(7C)	WAIT
128(80)	FBWT
132(84)	UDD
136(88)	RECADD
140(8C)	RECDEL
144(90)	BADD
148(94)	BDEL
152(98)	BCHG
156(9C)	IXADD
160(A0)	IXDEL
164(A4)	FINDS
168(A8)	SORTS
172(AC)	RECDS
176(B0)	STRECDS
180(B4)	DKAR
184(B8)	DKPR
188(BC)	DKRR
192(C0)	COMMITTS
196(C4)	BACKOUTS
200(C8)	UPDTTIME(MS)
204(CC)	LONGUPDTS

Table A-24. Offset location of SMF logout record statistics (Continued)

Offset dec(hex)	Statistic
208(D0)	LONGUPDTIME(MS)
212(D4)	SMPLS
216(D8)	RUNG
220(DC)	REDY
224(E0)	BLKI
228(E4)	WTSV
232(E8)	BLKO
236(EC)	SWPG
240(F0)	PCPU
244(F4)	DIRRCD
248(F8)	BXCHNG
252(FC)	BXDELE
256(100)	BXNEXT
260(104)	BXFIND
264(108)	BXINSE
268(10C)	BXSPLI
272(110)	BXFND
276(114)	BXFREE
280(118)	STCPU
284(11C)	STDEQ
288(120)	SCHDCPU
292(124)	SCREENS
296(128)	SVPAGES
300(12C)	PBRNFLT
304(130)	MQGETS
308(134)	MQPUTS
312(138)	UBUFHWS
316(13C)	MQHWTASK
320(140)	MQBYTEIN
324(144)	MQBYTEOU

Table A-24. Offset location of SMF logout record statistics (Continued)

Offset dec(hex)	Statistic
328(148)	MQHWQU
332(14C)	MQNUMQU
336(150)	MQNUMQM
340(154)	MQAPITIM
344(158)	MQAPICNT
348(15C)	MQGWTTIM
352(160)	MQGWTCNT
356(164)	MQGWTTSP
360(168)	MQGWTSUC
364(16C)	ECLOAD
368(170)	ECDELETE
372(174)	ECCALL
376(178)	ECCWAITM
380(17C)	ECCWAITS
384(180)	ECTWAITM
388(184)	ECTWAITS
392(188)	ECCTOUT
396(18C)	ECCNCT
400(190)	GTBLRU
404(194)	GTBLRS
408(198)	FSCBSW
412(19C)	Spare

SMF since-last record layout

Table A-25 shows the basic format of since-last Model 204 SMF records.

Table A-25. Since-last record basic format

Offset(hex)	Length	Description
2	2	Length of record
4	2	Zeros
6	1	System configuration

Table A-25. Since-last record basic format (Continued)

Offset(hex)	Length	Description
7	1	Record number (SMFSLRN or SMFLORN)
8	8	Time and date
16	4	System identification
20	8	Job name
28	8	Time and date of login
36	10	Login user ID
46	10	Login account
56	2	Binary user number
58	2	Two bytes reserved
60(3C)	4	Since-last type (EVAL CMPL DUMP REST...)
64(40)	8	File name
72(48)	10	Since-last subsystem name
82(52)	40	Since-last procedure name
122(7A)	2	Reserved two bytes
124(7C)	468	Binary fullwords of statistics (if CFRJRNL=0)
124(7C)	484	Binary fullwords of statistics (if CFRJRNL=1)

- Statistics depend upon the record number.
- SMF system identification field (offset 16) is the site-definable 4-character field retrieved from the SMF control area.
- SMF control area field contains the CPU identification string that is stored at system IPL time, and can be used to determine on which CPU Model 204 is running in a multi-CPU environment.

Offsets for SMF since-last record statistics

Table A-26 on page 598 shows the layout of SMF since-last record statistics. Each statistic in the table is: Length, 4; Data type, Binary. This record has the same statistics in the same order as the CCAJRNL user since-last records; only the offsets are different. For a description of each statistic, see Table A-1 on page 548.

Table A-26. SMF since-last record statistics

Offset dec(hex)	Description
124(7C)	NTBL

Table A-26. SMF since-last record statistics (Continued)

Offset dec(hex)	Description
128(80)	GTBL
132(84)	QTBL
136(88)	STBL
140(8C)	TTBL
144(90)	VTBL
148(94)	PDL
152(98)	FTBL
156(9C)	XTBL
160(A0)	ITBL
164(A4)	FSCB
168(A8)	OUTPB
172(AC)	HEAP
176(B0)	SQLI
180(B4)	SQLO
184(B8)	CNCT
188(BC)	CPU
192(C0)	DKRD
196(C4)	DKWR
200(C8)	UDD
204(CC)	OUT
208(D0)	SLIC
212(D4)	IN
216(D8)	RECADD
220(DC)	RECDEL
224(E0)	BADD
228(E4)	BDEL
232(E8)	BCHG
236(EC)	IXADD
240(F0)	IXDEL
244(F4)	FINDS

Table A-26. SMF since-last record statistics (Continued)

Offset dec(hex)	Description
248(F8)	SORT
252(FC)	RECDS
256(100)	STRECDS
260(104)	PCPU
264(108)	RQTM
268(10C)	DIRRCD
272(110)	BXCHNG
276(114)	BXDELE
280(118)	BXNEXT
284(11C)	BXFIND
288(120)	BXINSE
292(124)	BXSPLI
296(128)	BXRFND
300(12C)	BXFREE
304(130)	STCPU
308(134)	STDEQ
312(138)	SCHDCPU
316(13C)	SCREENS
320(140)	SVRD
324(144)	SVWR
328(148)	DKPR
332(14C)	SVPAGES
336(150)	COMMITTS
340(154)	BACKOUTS
344(158)	UPDTIME(MS)
348(15C)	LONGUPDTS
352(160)	LONGUPDTIME(MS)
356(164)	MQGETS
360(168)	MQPUTS
364(16C)	UBUFHWS

Table A-26. SMF since-last record statistics (Continued)

Offset dec(hex)	Description
368(170)	MQHWTASK
372(174)	MQBYTEIN
376(178)	MQBYTEOU
380(17C)	MQHWQU
384(180)	MQNUMQU
388(184)	MQNUMQM
392(188)	MQAPITIM
396(18C)	MAPICNT
400(190)	MQGWTTIM
404(194)	MQGWTCNT
408(198)	MQGWTTSP
412(19C)	MQGWTSUC
416(1A0)	ECLOAD
420(1A4)	ECDELETE
424(1A8)	ECCALL
428(1AC)	ECCWAITM
432(1B0)	ECCWAITS
436(1B4)	ECTWAITM
440(1B8)	ECTWAITS
444(1BC)	ECCTOUT
448(1C0)	ECCNCT
452(1C4)	GTBLRU
456(1C8)	GTBLRS
460(1CC)	FSCBSW
464(1D0)	Unused

Table A-27 lists the conflict statistics, if CFRJRNL=1.

Table A-27. Conflict statistics, if CFRJRNL=1

Offset dec(hex)	Description
468(1D4)	BLKCFRE
472(1D8)	BLKRLK
476(1DC)	WTCFR
480(1E0)	WTRLK

B

Coding SNA Communications Server Conversion Exit Routines

In this appendix

- Overview
- Rules governing data conversion exit routines
- X3270CHK (check for terminal characteristics)
- X3270OUT (convert output from 3270 format)
- X3270IN (convert input to 3270 format)

Overview

Exit routines that perform data conversion external to the Model 204 SNA Communications Server (formerly VTAM) 3270 Interface provide support for full-screen SNA Communications Server terminals that are not 3270 compatible.

This appendix explains the rules for coding, checking, and using these routines.

Rules governing data conversion exit routines

The following rules apply when you convert exit routines:

- Exit routines must acquire and release their own work areas.

These work areas are allocated out of the storage reserved by SPCORE for 24-bit addressing systems. If z/OS is available, the exit routines might allocate space above the 16 megabyte line.

- Exit routines must be serially reusable.
- Exit routines must avoid issuing any I/O or WAIT macros that adversely affect the performance of the whole Model 204 Online job.
- Exit routines must not issue STAE, SPIE, ESTAE, or ESPIE macros.
- Exit routines must preserve addressing mode and program interrupt mode.

Note: The value of AMODE must be 31 to ensure compatibility with z/OS.

For addresses above the 16MB line, the correct addressing mode is set by the following instruction, which restores the addressing mode in effect when the exit first gained control:

```
BSM 0,R14
```

- Exit routines must be named:
 - X3270CHK (check for terminal characteristics)
 - X3270OUT (convert output from 3270 format)
 - X3270IN (convert input to 3270 format)
- Define each exit routine as a separate entry point or CSECT.
- Assemble or compile exit routines after you code them and link the object modules into the appropriate Model 204 load modules.

X3270CHK (check for terminal characteristics)

Information about the terminal that is attempting to establish a session with Model 204 is passed to the X3270CHK routine. Provide the following information:

- Eight-character SNA Communications Server logical terminal identifier
- Terminal's session parameters
- SNA Communications Server LOGMSG associated with the terminal

The exit routine then returns a fullword that Model 204 passes to the conversion exit routines. The fullword can be used to determine various options, such as:

- Whether to convert the data streams for this terminal
- What terminal type is connecting to Model 204, allowing for the handling of more than one terminal type in the conversion exit routines

In addition to setting a value to the informational fullword, this exit routine can modify the SNA Communications Server LOGMSG associated with the

terminal or move in a message to use as the SNA Communications Server LOGMSG.

Communication protocols

The X3270CHK routine communicates with the Model 204 SNA Communications Server 3270 Interface in the following manner.

Upon entry

Register 1 — Contains the address of a parameter list in the following format:

- The first word contains the address of a fullword that holds an error return code value when leaving the exit routine.
- The second word contains the address of the 8-character SNA Communications Server logical terminal identifier.
- The third word contains the address of the session parameters associated with the terminal.
- The fourth word contains the address of the SNA Communications Server LOGMSG associated with the terminal. The fourth word always holds a valid address, even if there is no SNA Communications Server LOGMSG.
- The fifth word contains the address of a fullword that stores the maximum allowed length of the SNA Communications Server LOGMSG.
- The sixth word contains the address of a fullword that holds the current length of the SNA Communications Server LOGMSG. A length of zero means there was no SNA Communications Server LOGMSG.
- The seventh word contains the address of a fullword that has information to pass to the conversion exit routines. The initial value of the fullword is zero. The high-order bit of this word is on to indicate the end of the parameter list.

Register 13 — Contains the address of an 18 fullword register savearea.

Register 14 — Contains the address in the Model 204 SNA Communications Server 3270 Interface to which the X3270CHK exit routine must branch when it finishes processing.

Register 15 — Contains the address of the X3270CHK exit routine.

Registers 2–12 must be preserved by the X3270CHK exit routine for Model 204.

Upon leaving

The arguments pointed to by the parameter list can be modified by the exit routine in the following ways:

- Set the error return code field. It cannot be assumed that it is set to zero upon entry. A value of zero indicates that the exit routine had no problem checking for the terminal characteristics it was looking for. If the value is nonzero, the Model 204 SNA Communications Server 3270 Interface does the following:
 - Issues an error message about a problem with the terminal, giving the terminal's logical SNA Communications Server identifier and the error return code value to the audit trail.
 - Issues a CLSDST to the terminal to terminate the terminal's request for a session. The fullword to be passed to the conversion exit routines should be set. If the SNA Communications Server LOGMSG is modified, the current length argument must be updated.
- If the exit routine is written in Assembler Language, the copy member X3270CPL in the Rocket Software-supplied macro library can be used to define the routine's parameter list. The labels used in this copy member are as follows:
 - X3270CPL is the DSECT name.
 - X3270CRC is the address of the return code field.
 - X3270CID is the address of the terminal's logical network ID.
 - X3270CSP is the address of the terminal's session parameters.
 - X3270CLG is the address of the terminal's SNA Communications Server LOGMSG.
 - X3270CML is the address of the maximum SNA Communications Server LOGMSG length.
 - X3270CLL is the address of the current SNA Communications Server LOGMSG length.
 - X3270CCW is the address of the word to be generated by X3270CHK.

X3270OUT (convert output from 3270 format)

The X3270OUT routine is responsible for converting an output data stream in 3270 format from Model 204 to the appropriate device protocol. The exit routine is given the address of the actual output buffer that Model 204 passes to SNA Communications Server, into which the converted output data stream must be placed. This exit routine must be able to handle all the 3270 output commands and attributes.

If this conversion exit is linked into the Model 204 Online job, it is then responsible for moving the output data into the Model 204 SNA Communications Server output buffer even if no actual conversion is performed.

The X3270OUT routine communicates with the Model 204 SNA Communications Server 3270 Interface in the following manner.

Upon entry

Register 1 — Contains the address of a parameter list in the following format:

- The first word contains the address of a fullword that encloses a return code value when leaving the exit routine.
- The second word includes the address of the fullword generated by the exit routine, X3270CHK.
- The third word contains the address of a Model 204 SNA Communications Server output buffer. The converted output data stream must occupy the SNA Communications Server output buffer upon leaving the X3270OUT routine.
- The fourth word contains the address of the fullword that includes the length of the Model 204 SNA Communications Server output buffer. The fourth word corresponds to the Model 204 parameter, LOU TPB, on the first SNA Communications Server 3270 thread. The length of the converted output data stream must not exceed the LOU TPB value.
- The fifth word includes the address of the 3270 output data stream.
- The sixth word contains the address of the fullword enclosing the current length of the 3270 output data stream. The high-order bit of this word is on to indicate the end of the parameter list.

Register 13 — Contains the address of an 18-fullword register savearea.

Register 14 — Contains the address in the Model 204 SNA Communications Server 3270 Interface to which the X3270OUT exit routine must branch when it finishes processing.

Register 15 — Contains the address of the X3270OUT exit routine.

Registers 2–12 must be preserved by the X3270OUT exit routine for Model 204.

Upon leaving

The following arguments pointed to by the parameter list must be modified by the exit routine:

- The return code field must be set. (It cannot be assumed that it is set to zero upon entry.) A value of zero indicates that the data stream conversion was successful. If the value is nonzero, the Model 204 SNA Communications Server 3270 Interface does the following:
 - Issues an error message stating the failure of the output data stream conversion, with the return code value, to the audit trail.
 - Takes a SNAP of the user's thread for diagnostic information.
 - Performs a soft restart of the user's thread. The length of the converted output data stream field must be set in the current output length argu-

ment.

- If the exit routine is written in Assembler Language, the copy member X3270OPL in the Rocket Software-supplied macro library can be used to define the routine's parameter list. The labels used in this copy member are as follows:
 - X3270OPL is the DSECT name.
 - X3270ORC is the address of the return code field.
 - X3270OCW is the address of the word generated by X3270CHK.
 - X3270OBF is the address of the Model 204 SNA Communications Server output buffer.
 - X3270OML is the address of the maximum SNA Communications Server buffer length.
 - X3270ODA is the address of the 3270 output data stream.
 - X3270ODL is the address of the 3270 output data stream length.

X3270IN (convert input to 3270 format)

The X3270IN exit routine is responsible for converting an input data stream from the appropriate device protocol to 3270 format for input to Model 204.

Communications protocols

The X3270IN routine communicates with the Model 204 SNA Communications Server 3270 Interface in the following manner.

Upon entry

Register 1 — Contains the address of a parameter list in the following format:

- The first word contains the address of a fullword that includes a return code value when leaving the exit routine.
- The second word contains the address of the fullword generated by the exit routine X3270CHK.
- The third word contains the address of a Model 204 SNA Communications Server input buffer. The converted input data stream must occupy this buffer upon leaving this routine.
- The fourth word contains the address of a fullword that holds the length of the input data stream. The length of the converted 3270 input data stream is passed back in this argument.
- The fifth word contains the address of a fullword that holds the length of the Model 204 SNA Communications Server input buffer. This corresponds to the Model 204 LOU TPB parameter on the first SNA Communications Server 3270 thread. The length of the converted input data stream must not

exceed the LOU TPB value. The high-order bit of this word is on to indicate the end of the parameter list.

Register 13 — Contains the address of an 18-fullword register savearea.

Register 14 — Contains the address in the Model 204 SNA Communications Server 3270 Interface to which the X3270IN exit routine must branch when it finishes processing.

Register 15 — Contains the address of the X3270IN exit routine.

Registers 2-12 must be preserved by the X3270IN exit routine for Model 204.

Upon leaving

The following arguments pointed to by the parameter list must be modified by the exit routine:

- The return code field must be set. (Do not assume that it is set to zero upon entry.) A value of zero indicates that the conversion of the data stream was successful. If the value is nonzero, the Model 204 SNA Communications Server 3270 Interface does the following:
 - Issues an error message about the failure of the conversion of the input data stream, with the return code value to the audit trail.
 - Takes a SNAP of the user's thread for diagnostic information.
 - Performs a soft restart of the user's thread. The length of the converted 3270 input data stream must be set in the argument that held the length of the original input data stream.
- If the exit routine is written in Assembler Language, the copy member X3270IPL in the Rocket Software-supplied macro library can be used to define the routine's parameter list. The labels used in this copy member are as follows:
 - X3270IPL is the DSECT name.
 - X3270IRC is the address of the return code field.
 - X3270ICW is the address of the word generated by X3270CHK.
 - X3270IBF is the address of the Model 204 SNA Communications Server input buffer.
 - X3270IDL is the address of the input data length.
 - X3270IML is the address of the maximum SNA Communications Server input buffer length.

X3270IN (convert input to 3270 format)

C

ALLOCATE Utility in z/VSE

In this appendix

- Overview
- ALLOCATE control statement
- ALLOCATE z/VSE job stream

Overview

The ALLOCATE utility provided with MODEL 204 allocates Model 204 files, including CCATEMP, CCAGRP, and CCASERVER, in the z/VSE environment. One or more files, as specified in control statements, can be allocated during one execution. You must provide a DLBL and EXTENT with complete information for each file name referenced in the ALLOCATE control statement.

ALLOCATE control statement

The ALLOCATE control statement format is:

```
ALLOCATE FILE (filename1 filename2 filenamex)
```

The following considerations apply:

- ALLOCATE statement is free format and can begin in any column.
- You can include any number of ALLOCATE control statements in the input to the utility.

- Use a continuation character (-) after the last parameter on an input record being continued.
- There is no limitation to the number of continuation statements.
- *filename1* through *filenamex* refer to the file names on the DLBL statements in the job control stream.
- If a file name referenced in the ALLOCATE statement control does not have a corresponding DLBL statement in the JCL, an error message is logged in the output audit trail.

ALLOCATE z/VSE job stream

The following job stream runs the ALLOCATE utility:

```
// JOB ALLOCATE MODEL 204 FILES
// DLBL M204CL,'M204.CORE.IMAGE.LIBRARY'
// EXTENT ,volser
// LIBDEF CL,SEARCH=M204CL
// DLBL CCATEMP,'CCATEMP.SCRATCH.FILE',0
// EXTENT SYS001,SYSWK1,,,1000,390
// DLBL CLAIM81,'DEMO.CLAIMS81',99/365
// EXTENT SYS001,SYSWK1,,,1390,780
// DLBL CLAIM82,'DEMO.CLAIMS82',,,,99/365
// EXTENT SYS001,SYSWK1,,,1780,780
// DLBL CLIENTS,'DEMO.CLIENTS',99/365
// EXTENT SYS001,SYSWK1,,,3340,780
// DLBL VEHICLE,'DEMO.VEHICLES',99/365
// EXTENT SYS001,SYSWK1,,,4120,780
// ASSGN SYS001,DISK,VOL=SYSWK1,SHR
// EXEC ALLOCATE,SIZE=AUTO
*ALLOCATE DEMONSTRATION DATABASE FILES
ALLOCATE FILE (CLAIM81)
ALLOCATE FILE (CCATEMP -
CLAIM82)
ALLOCATE FILE (VEHICLE,CLIENTS)
/*
/&
```

Index

Symbols

\$CODE function 241
\$CURFILE function
 STBL requirements 59
\$DECODE function 241
\$FLCFILE function
 STBL requirements 59
\$INCRG function 52
\$POST('QZSIG') function call
 limitation 392
\$SETG function 52
\$STATUS return codes
 ECF 526
\$STATUSD return codes
 ECF 526
\$UPDATE function
 STBL requirements 59
\$UPDFILE function
 STBL requirements 59
*DEVICE command 112
*SLEEP command 112
*SNAP command 113

Numerics

31-bit processing 6, 39
 asynchronous checkpoint 334
 IFAM2 and CICS 92
 pseudo subtasks 333
31-bit storage 29, 38, 39
 using 39
3270 terminals
 IODEV=7 72
3767 terminals
 IODEV=37 72
64-bit mode 243
64-bit statistics 9, 46, 149
64-bit storage 38
 not available on z/VSE 39
 using 39

A

abend code for dumps 319
above the bar
 NUMBUFG setting for buffers 40
 screens and images 42
 storage 42
above the bar storage
 and EBM pages 41
 CCATEMP 8
 virtual 8
 z/Architecture 8
Access methods 65, 609
 BSAM 67
 CRAM 74, 75
 sequential processing 512
 VTAM 72, 116, 603
ACCTIM parameter 576
active subsystem
 leaving unchanged 226
 updating 226
Active Subsystem Help, using 226
ad hoc file groups 157
AD message type 321
adding security to files 272
addressing mode 604
AGEINCR scheduling parameter 141
AGEINTVL scheduling parameter 141
AGESCAN scheduling parameter 141
AGESLICE scheduling parameter 141
ALLOCATE
 job stream (z/VSE) 612
ALLOCATE command 498
 description 113
 migrated dataset 71
 obsolete form 498
 with XTLOT or TIOT 498
ALLOCATE control statement
 format 611
ALLOCATE utility 611
allocating
 Model 204 files 611
 server datasets 171
ALTIODEV=45 105
ALTIODEV=47 105

- AMODE option 604
- Analytics/204
 - description of 11
- ANALYZE command
 - syntax for 308
- application languages
 - interfaces to 479
 - using HLI with 479
- APSY
 - load statistics 180
 - precompiled procedures in Storage feature 178
 - See SUBSYSMGMT
- APSY precompiled procedures
 - storage requirements 179
- APSY Precompiled Procedures in Storage
 - DSOPT parameter 180
- APSY support
 - dynamic 224
- APSYLD statistic
 - description and offset 548
 - tracking CCASYS procedures 181
- APSYLDD statistic
 - description and offset 548
 - tracking dataspace loads 181
- APSYLDT statistic
 - description and offset 548
 - tracking tiny loads 181
- APSYPAGE parameter
 - activating APSY Precompiled Procedures in Storage 178
 - compared to TEMPPAGE 178
- Archived datasets
 - recalling 71
- archiving data 157
- Assembler application language
 - with Model 204 HLI 479
- asynchronous checkpoint 334
- Asynchronous SVC dumps 143
- AT MOST ONE attribute
 - and transaction back out 356
- attributes
 - changes in dynamic APSY subsystem 224
- Audience xxv
- AUDIT statistic
 - description and offset 548
- audit trail
 - and update IDs 363
 - and z/VM 306
 - and z/VSE 305
 - boundaries of 364
 - recovery messages 368
 - tracking update units 363
- AUDIT204 commands
 - FORMAT 309
- AUDIT204 utility
 - ANALYZE command 308
 - CCAIN statement 307
 - EXEC (CMS) 317
 - FORMAT command 309
 - JCL (z/OS) 313
 - JCL (z/VSE) 316
 - processing multiple tapes 318
 - purpose of 306
 - REPORT command 310
 - syntax for 317
- AUDITxx options
 - MSGCTL command 323
 - processing and hierarchy 323
- automatic COMMIT 204
- automatic login
 - subsystem 204
- automatic logout 204
- automatic member 182
- automatic start 203
- AUTOSYS parameter 191
 - invoking an individual subsystem 196
 - user environment control 105

B

- back out log
 - compensating updates 354
 - definition of 355
- back out records
 - and ROLL FORWARD processing 365
- backing out update activity
 - transaction back out 352
- backing up files
 - media recovery 349
- BACKOUTS statistic
 - description and offset 548
- BADD statistic
 - description and offset 548
- BATCH2
 - *DEVICE command 112
 - *SLEEP command 112
 - backup requirements 392
 - IODEV required for backup 392
- BATCH2 utility
 - definition of 490
- BATCH204 jobs
 - archived datasets 71
 - CCATEMP file 153
 - configuration 5
 - sample z/OS JCL 23
- BCHG statistic
 - description and offset 548

- BDEL statistic
 - description and offset 548
- BEFORE clause
 - of REGENERATE command 383
- before-images, See ROLL BACK facility
- below the bar
 - freeing buffer space 40
- BLKCFRE statistic
 - description and offset 548
- BLKI statistic
 - description and offset 549
- BLKO statistic
 - description and offset 549
- BLKRLK statistic
 - description and offset 549
- block size
 - sequential processing 513
- BROADCAST command 113
- BSAM 67
- buffer allocation
 - BSAM 71
 - z/OS CRAM 85
- buffer overruns
 - detecting 40
- buffer pool data structures
 - hash cells 38
- buffer pools
 - above the bar 40, 42
- buffers 24
 - adequacy of 436
 - disk 37
 - for UTILJ REPORT option 396
 - in pseudo conversational CICS 543
 - monitoring of 435
 - sequential processing requirement 512
- BUMP command 113
 - and EXTERNAL CALL statement 521
 - example code 521
 - with FORCE option 521
- BXCHNG statistic
 - description and offset 549
- BXDELE statistic
 - description and offset 549
- BXFINND statistic
 - description and offset 549
- BXFREE statistic
 - description and offset 549
- BXINSE statistic
 - description and offset 549
- BXNEXT statistic
 - description and offset 549
- BXRFND statistic
 - description and offset 549
- BXSPLI statistic

- description and offset 549
- BYPASS parameter 19

C

- cache fast write 171, 450
- cache hiperspace
 - DSOPT parameter 180
- CACHE parameter 450
- call name
 - associating an external program 525
 - definition of 520
- CANCEL command 117
- CAUDIT parameter
 - values of 299
- CAUDIT user parameter 105
- CCA Analytics
 - see Analytics/204 11
- CCAAUDIT file
 - CAUDIT parameter 299
 - format, line 303
 - generation of 305
 - in CMS 306
 - in DOS 305
 - LAUDIT parameter 300
 - LAUDPROC parameter 305
 - lines, types of 304
 - AD 304
 - CI 304
 - CP 304
 - CS 304
 - ER 304
 - LI 304
 - LP 304
 - LR 304
 - LS 304
 - MS 304
 - OI 304
 - OO 304
 - QT 305
 - RK 305
 - ST 305
 - US 305
 - XX 305
 - logical input control 300
 - output control 305
 - physical input control 299
 - procedure name length 305
 - SYSOPT parameter 305
- CCAGEN file
 - with REGEN IGNORE 384
- CCAGRP file 159
 - and SYSOPT settings 161

- copy of 113
- creation of 113
 - CREATEG 113
- data integrity 34
- storing file definitions 161
- CCAIN dataset
 - required for media recovery 386
- CCAIN input stream
 - AUDIT204 307
 - BATCH204 23
 - runtime environment 14
 - structure 14
 - system control 144
 - user parameters 64
 - z/VM 26
 - z/VSE 25
- CCAJLOG
 - defining a stream 295
 - switching streams 288
- CCAJRNL
 - and CCAJLOG 297
 - stream configurations 290
 - switching streams 288
- CCAJRNL file 293, 371
 - and CCAAUDIT 293
 - BLKSIZE parameter 300
 - contents of 293
 - creation of 287
 - entries 298, 560, 562, 563
 - categories of 298
 - format of 298
 - header format 560
 - recovery 562
 - system statistics 563
 - trailer format 562
 - error messages 299
 - file statistics 586, 587
 - final 586
 - partial 587
 - format of 298
 - FRCVOPT parameter 292
 - initialization statistics format 590
 - logging individual update 361
 - merged journal bracketing statistics format 591
 - monitoring statistics 302
 - multiple journal statements 288
 - NJBUFF parameter 300
 - output control 299
 - printing of 294
 - RCVOPT parameter 292
 - ROLL FORWARD processing 360
 - specifying roll forward logging 292
 - statistics 294
 - printing of 301
 - storage representation 298
 - subtypes 302
 - statistics lines 563, 586, 587
 - file 586
 - system 563
 - storing update records 287
 - SYSOPT parameter 299
 - system final statistics 563, 564
 - system partial statistics 563, 564
 - system performance statistics 563
 - system statistics 563
 - system statistics format 564, 571, 572
 - disk buffer monitor 572
 - performance 571
 - text entry statistics format 588, 589
 - non-continued 588
 - time stamp statistics format 591
 - use of 294
 - user statistics 575, 576, 580, 585
 - final 575
 - partial 576
 - performance 585
 - since-last 580
 - user statistics format 576, 581, 584, 585, 586
 - logout 576
 - partial 576
 - performance 586
 - since-last 581, 584, 585
- CCALL entry points 239
- CCAMDMP datasets
 - coordinated with CCASNAP 326
 - DCB characteristics 326
 - estimating space required 326
 - number of 326
 - processing 327
 - rules for using 325
- CCAPRINT dataset
 - required for media recovery 386
- CCAPRINT file
 - in CMS 283
 - in z/OS 282
 - in z/VSE 282
- CCARF dataset
 - NOTE/POINT facility 381
 - ROLL FORWARD processing 379
- CCARF file 371
- CCASERVER file
 - data integrity 34
- CCASERVER in Storage
 - activating 167
 - DESPOPT parameter 180
 - managing server swapping 151, 166
 - system programmers consideration 151, 167
- CCASNAP dataset

- coordinating with CCAMDMP 326
- CCASnap file
 - CMS 324
 - MSGCTL command 320
 - SNAPCTL parameter 320
 - z/OS requirements 324
 - z/VSE 324
- CCASTAT
 - creating a backup 259
 - deleting entries 259
 - modifying with ZCTLTAB 264
 - updating with Password Expiration feature 260
 - updating with ZCTLTAB 262
- CCASTAT file
 - BSAM 515
 - creating a password table 249
 - creation of 249
 - creation of in z/OS 250
 - creation of in z/VM or CMS 250
 - creation of in z/VSE 250
 - data integrity 34
 - DD statement 251
 - LOGFILE command 115
 - login security 252, 256
 - LOGKEY command 115
 - LOGLST command 115
- CCASTAT messages
 - managing 267
- CCASYS file 181
 - and DICTIONARY 176
 - and SUBSYSMGMT 176
 - and the Subsystem Management facility 176
 - creation of 176
 - DD statement 177
 - enabling of 177
 - recovery of 178
 - reorganizing 177
 - statistics 176
 - SUBSYSMGMT 181
 - subsystem requirements 177
 - SYSOPT parameter 177
- CCASYS procedures
 - tracking with APSYLD 181
- CCATEMP file 365
 - above the bar 8
 - batch run 147
 - constraints log 354
 - data integrity 34
 - DOS JCL 152
 - file groups 147
 - FLOD exit 147
 - IFAM call 147
 - in z/VSE 152
 - multiple jobs 149
 - ONLINE 147
 - precompiled requests 147
 - Subsystem Management facility 147
 - transaction back out facility 147
 - transaction back outs 354
 - User Language request 147
 - z/OS JCL 150
- CCATEMP in Storage
 - activating 151
 - managing server swapping 151, 166
 - system programmers consideration 151, 167
- CCATEMP pages
 - CDMAXP2X parameter 356
 - CDMINP2X parameter 356
- CDLWAIT statistic
 - description and offset 549
- CDMAXP2X parameter
 - number of CCATEMP pages kept in compacted form 356
- CDMINP2X parameter
 - number of permanently allocated CCATEMP pages 356
- CDTB module
 - CODE macro 242
- CFRJRNL parameter 584
- CFRLOOK parameter 584
 - specifies 27
- CFRWPCT scheduling parameter 141
- character string table (STBL) 58
- checkpoint
 - aborting 113, 335
 - asynchronous 334
 - CHECKPOINT command 333, 335
 - CHKABORT command 113
 - CHKMSG command 335
 - CHKPOINT file 340, 342
 - facility 332
 - file processing facility 410
 - IFCHKPNT call 333
 - IFFNSH call 333, 335
 - IFSTRT call 333, 335
 - logging 332, 340
 - M204CKPX ASSEMBLER user exit 336
 - parallel stream 425, 426
 - in CMS 426
 - in z/OS 425
 - in z/VSE 425
 - process of 334
 - record trailer 343, 344
 - record type codes 343
 - table identifier codes 344
 - record types 343
 - status 335
 - stream 411, 416, 417

- concatenated 416
 - parallel 417
 - ring 411
 - update units 334
 - UTILC utility 342, 344, 346
 - JCL 344
- CHECKPOINT command 333, 335
- CHECKPOINT END EXTENDED QUIESCE command
 - terminating extended quiesce state 389
- checkpoint extended quiesce
 - introducing 389
- CHECKPOINT SET EXTENDED QUIESCE command
 - beginning extended quiesce 389
- CHECKPOINT UNSET EXTENDED QUIESCE command
 - reversing the SET option 389
- checkpoints
 - automated functionality 290
 - finding quickly 381
 - M204CKPX user exit 335
- CHKABORT command 335
- CHKAWW pseudo subtask 334
- CHKMSG command 335
- CHKP module 360
- CHKPNTD dataset
 - and z/VSE 378
- CHKPNTS dataset
 - creating 341
- CHKPNTS, switching streams 289
- CHKPOINT dataset
 - discontinuity records 363
 - logging preimages 351
 - required for recovery 378
 - required for ROLL BACK processing 379
 - resolving file discontinuities 363
 - sizing with CPTIME 342
- CHKPOINT file 371
 - creation of 341
- CHKPOINT, switching streams 289
- CHKPPST pseudo subtask 44
- CICS
 - output routing 502
- CKD devices 168, 169
- CLEAR GLOBALS statement 52
- CLEAR command 52
- CLEARGO command 52
- client subsystem
 - access 177
- CMS
 - ALLOCATE command 113, 498
 - CCAAUDIT file 306
 - CCAPRINT file 283
 - CCASNAP file 324
 - CCASTAT, creation of 250
 - DEFINE PRINTER command 502
 - DEFINE PUNCH command 503
 - dumps 324
 - file group dataset 160
 - M204 command 101
 - parallel checkpoint stream 426
 - parallel stream FILEDEF 426
 - ring/parallel journal stream 424
 - sequential processing 512
 - System Management Facility (SMF) 592
 - tape mounts 22
 - User 0 283
 - user zero
 - output 283
 - UTILC utility 345
 - VTAM interface 73
- CMS operating system
 - MERGEJ utility code 406
 - ONLINE command 19
 - utilities 20
 - UTILJ utility code 399
- CMS-format disks 10
- CNCT statistic
 - description and offset 549
- COBOL application language
 - with Model 204 HLI 479
- CODE macro 242
- Command file facility 103
- Commands
 - ONLINE 19
 - REPORT 310
 - RESTOREG 161
- commands
 - *DEVICE 112
 - *SLEEP 112
 - *SNAP 112
 - ALLOCATE 112
 - AUTHCTL 112
 - BROADCAST 112
 - BUMP 112
 - CANCEL 117
 - CHECKPOINT 333, 335
 - CHKABORT 112
 - CHKMSG 335
 - COMMIT
 - automatic 204
 - COPY 112, 410, 414
 - CREATE 112, 159, 274
 - CREATE GROUP 273
 - CREATEG 112, 159
 - DEFINE 113
 - DEFINE DATASET (z/OS) 493

- DEFINE DATASET (z/VSE) 495
- DEFINE FIELD 274
- DEFINE LINK 545
- DEFINE PRINTER 502
- DEFINE PROCESS 545
- DEFINE PROCESSGROUP 545
- DEFINE PUNCH 503
- DEFINE STREAM 288, 410, 413, 422, 426
 - AUTOOFFLOAD option 413
 - offload 426
 - recovery 422
 - use 288
- FREE 500
- IFAMFORCE 114
- IFAMHALT 114
- IFAMOPEN 114
- IFAMSTART 114
- IFAMSTAT 114
- IFRECV 542
- IFSENDX 542
- IFSGNL 542
- INITIALIZE 274
- LOGCTL 114, 255
- LOGFILE 115, 272
- LOGGRP 115, 273
- LOGKEY 115
- LOGLST 115, 254
- MONITOR 115
- MONITOR SUBSYSTEM 458
- MSG 115
- MSGCTL 115, 299
 - use 299
- OFFLOAD 115, 413
- OPENCTL 270, 274
- operator 119
- PRIORITY 116
- REACTIVATE 116
- REGENERATE 116, 351, 383
- RESTART 351
- RESTOREG 116
- START 116
- SUBSYSMGMT
 - CREATE GROUP 208
 - START FILE 195
 - START SUBSYSTEM 195
 - STOP FILE 195
 - STOP SUBSYSTEM 195
 - TEST 195
- SUBSYSMGMT DEBUG SUBSYSTEM 195
- SUBSYSMGMT START SUBSYSTEM 195
- system control 112
- TMASKUPDATE 116
- USE 502
- USE PRINTER 504
- USE PUNCH 505
- VIEW 116
- VTAMOFF 116
- WARN 116
- commit
 - automatic 204
 - definition of 352
- COMMIT RELEASE statement 352
- COMMIT statement
 - lock-pending-updates 354
- COMMITTS statistic
 - description and offset 549
- Common Service Area (CSA) 84
- Communications global variable
 - subsystems 188
- compatibility issues
 - CCASTAT going back 267
 - subsystems at initialization 177
- Compensating updates
 - backing out incomplete transactions 354
- compiler variable table (VTBL) 59
- concatenated stream
 - description 416
 - example 426
- concatenated streams
 - SWITCH STREAM command 291
- condition codes for ZCTLTAB 261
- Connect*
 - description of 11
- console communication (z/VSE) 131
- constraints database
 - understanding 355
- constraints log 365
 - CCATEMP space 354
 - optimizing performance 355
 - size 355
- Conventions, notation xxv
- converser mode
 - CRAM 83
- COPY command 410
 - description 113
- COUNT option 320
- CPMAX parameter
 - Checkpoint facility 333
 - restrictions 411
 - saving checkpoints 342
 - specifies 27
 - used to size CHKPOINT 342
- CPQZ
 - posting and unposting automatically 389
- CPQZSECS parameter
 - managing an extended quiesce 392
- CPSORT parameter
 - Checkpoint facility 333

- CPTIME parameter
 - Checkpoint facility 333
 - sizing CHKPOINT dataset 342
 - specifies 27
- CPTO parameter
 - Checkpoint facility 333
- CPTQ parameter
 - Checkpoint facility 333
- CPTS parameter
 - Checkpoint facility 333
- CPTYPE
 - usage requirements 339
- CPTYPE parameter
 - Guaranteed Checkpoint facility 333
- CPU statistic
 - description and offset 549
- CPU time
 - saving with EXCPVR 433
- CPU types 6
- CPU usage
 - Timer PC 433
 - Timer SVC 433
- CPUSLICE scheduling parameter 141
- CRAM 74
 - allocation buffers for 86
 - channel names 83
 - choosing a type of 76
 - communicating with multiple regions 83
 - definition of 74
 - in converser mode 83
 - in z/OS and z/VSE 75
 - modes of operation 83
 - required for BATCH2 backup 392
 - sizing communication area 89
- CRAM channel name
 - TSO Interface 93
- CRAM master 84
- CRAM OPEN command 83
- CRAM subsystem
 - locating for z/VSE 87
 - partition requirements 88
- CRAMZWT subtask 88
- CREATE command 159, 274
 - description 113
- CREATE GROUP command 273
- CREATEG command 159
- Creating a Server Dataset (CCASERVER) 165
- creating system files
 - CHKPOINT 341
- CRFSCHNL parameter 83
- CRIO module
 - IFDIAL thread support 484
- CRIOCHNL parameter 83
 - defined for BATCH2 backup 392

- Cross-memory data mover. See XDM
- Cross-Partition Communications Services. See XPCC
- Cross-Partition Event Control Blocks. See XECB
- Cross-Region Access Method (CRAM)
 - performance 449
- Cross-Region Access Method. See CRAM
- CSA storage 84
- cyclic data
 - accessing 157

D

- DASD
 - cached controllers 450
 - shared 36
- DASD device types 6
- data integrity
 - and resource locking 34
 - encryption 249
- data length
 - sequential processing 513
- DATALEN parameter
 - Horizon SQL 455
- Dataset Control Blocks
 - and GDGs 420
- datasets
 - dynamically allocated, recovering 382
 - limits with TIOT 497
 - See files
 - See stream configurations
 - unlimited with XTIO 497
- dataspaces
 - definition of 151, 166
 - putting CCATEMP and CCASERVER in Storage 151, 166
 - tracking loads with APSYLD 181
- DATE routine 239
- DATE3 routine 239
- DATE4 routine 239
- DBIDs
 - and MAXSIMIO 435
 - calculating default number of 435
- DCB characteristics
 - of CCAMDMP datasets 326
- DD statements
 - required for IFAM1 480
- DEBUG SUBSYSTEM command 195
- debugging
 - SUBSYSMTGMT 212
- debugging, See *SNAP command
- deferred update mode 33
- deferred updates

- how recovery works 382
 - recovery feature 382
 - recovery for 381
 - ROLL FORWARD processing 382
 - z/VSE exclusion 382
- DEFINE command 14, 493, 502, 503
 - DEFINE FIELD, security 274
 - DEFINE LINK 545
 - DEFINE PRINTER 502
 - DEFINE PROCESS 545
 - DEFINE PROCESSGROUP 545
 - DEFINE PUNCH 503
 - DEFINE STREAM 288, 410, 422, 426
 - concatenated 426
 - ring/parallel 422
 - use 288
 - description 113
 - override of 502
 - TIOT and XTIO options 498
- DEFINE commands, Model 204
 - example of 96
- DEFINE DATASET command
 - migrated dataset 71
- DESECURE command
 - removing security from files 272
- DEV10 statistic
 - description and offset 550
- DEV11 statistic
 - description and offset 550
- DEV12 statistic
 - description and offset 550
- DEV13 statistic
 - description and offset 550
- DEV14 statistic
 - description and offset 550
- DEV17 statistic
 - description and offset 550
- DEV18 statistic
 - description and offset 550
- DEV19 statistic
 - description and offset 550
- DEV20 statistic
 - description and offset 550
- DEV23 statistic
 - description and offset 550
- DEV24 statistic
 - description and offset 550
- DEV27 statistic
 - description and offset 550
- DEV28 statistic
 - description and offset 550
- DEV31 statistic
 - description and offset 550
- DEV32 statistic
 - description and offset 550
- DEV37 statistic
 - description and offset 550
- DEV38 statistic
 - description and offset 550
- DEV49 statistic
 - description and offset 550
- DEV5 statistic
 - description and offset 550
- DEV50 statistic
 - description and offset 550
- DEV53 statistic
 - description and offset 550
- DEV54 statistic
 - description and offset 550
- DEV55 statistic
 - description and offset 550
- DEV56 statistic
 - description and offset 550
- DEV57 statistic
 - description and offset 551
- DEV58 statistic
 - description and offset 551
- DEV59 statistic
 - description and offset 551
- DEV6 statistic
 - description and offset 550
- DEV60 statistic
 - description and offset 551
- DEV61 statistic
 - description and offset 551
- DEV62 statistic
 - description and offset 551
- DEV63 statistic
 - description and offset 551
- DEV64 statistic
 - description and offset 551
- DEV65 statistic
 - description and offset 551
- DEV66 statistic
 - description and offset 551
- DEV67 statistic
 - description and offset 551
- DEV68 statistic
 - description and offset 551
- DEV69 statistic
 - description and offset 551
- DEV7 statistic
 - description and offset 550
- DEV70 statistic
 - description and offset 551
- DEV71 statistic
 - description and offset 551
- DEV72 statistic

- description and offset 551
- DEV73 statistic
 - description and offset 551
- DEV74 statistic
 - description and offset 551
- DEV8 statistic
 - description and offset 550
- DEV9 statistic
 - description and offset 550
- device types
 - z/OS 65
 - z/VM 66
 - z/VSE 65
- DICTIONARY 181
- Dictionary files
 - reorganizing 177
- dictionary, in-core 208
- Dictionary/204
 - data definition errors 227
- directed output
 - CCAPU dataset 503
 - CICS 502
 - CMS 503
 - M204SPL EXEC 503
 - DEFINE PRINTER command 502
 - DEFINE PUNCH command 503
 - destinations of 501
 - USE command 502
 - USE PRINTER command 504
 - USE PUNCH command 505
 - z/OS 502
 - CCAPR dataset 502
 - z/VSE
 - CCAPPR dataset 507
 - z/VSE/POWER 507
- DIRRCD statistic
 - description and offset 551
- DISABLE SUBSYSTEM FILE command 184
- disabling
 - transaction back out 356
- discontinuities
 - files and ROLL FORWARD processing 364
 - how they occur 363
 - resolving 363
- discontinuity record
 - CHKPOINT dataset 363
- Disk Buffer I/O control blocks (DBID)
 - calculating default number of 435
- disk buffer overrun detection
 - purpose of 38
- disk buffer pool
 - used for 37
- disk buffers 24
 - usage and managing 37
- disk devices
 - servers on 167
- disk updates 43
- disks
 - CMS-format 10
 - variable-format 10
- DISPDUPS option
 - UTILJ utility 394
- DISPLAY command
 - Early Warnings 282
- Distributed Application Facility (DAF) 542
- DKAR statistic
 - description and offset 551
- DKPR statistic
 - description and offset 551
- DKPRF statistic
 - description and offset 551
 - fast logical page reads 461
 - tracking fast reads 461
- DKRD statistic
 - description and offset 551
- DKRDL statistic
 - description 551
- DKRR statistic
 - description and offset 551
- DKSAWB statistic
 - description and offset 551
- DKSAWBL statistic
 - description 552
- DKSAWW statistic
 - description and offset 552
- DKSAWWL statistic
 - description 552
- DKSDIR statistic
 - description and offset 552
- DKSDIRT statistic
 - description and offset 552
- DKSFBS statistic
 - description and offset 552
- DKSKIP statistic
 - description and offset 552
- DKSKIPT statistic
 - description and offset 552
- DKSRR statistic
 - description and offset 552
- DKSRRFND statistic
 - description and offset 552
- DKSTBLA statistic
 - description and offset 552
- DKSTBLB statistic
 - description and offset 553
- DKSTBLC statistic
 - description and offset 553
- DKSTBLD statistic

- description and offset 553
- DKSTBLF statistic
 - description and offset 553
- DKSTKQC statistic
 - description and offset 553
- DKSWRP statistic
 - description and offset 553
- DKSWRPT statistic
 - description and offset 553
- DKUPDTWT parameter 43
- DKUPTIME statistic
 - description and offset 553
- DKWR statistic
 - description and offset 553
- DKWRL statistic
 - description 553
- double word statistics 9
- DSPOPT parameter
 - moving saved compilation pages 179
 - using expanded storage 180
 - with APSY Precompiled Procedures in Storage 180
 - with CCASERVR in Storage 180
- Dummy string and \$READ table (ITBL) 53
- dummy strings
 - in precompilable procedures 234
- DUMP statistic
 - description and offset 553
- DUMPG command
 - back up group definitions 161
- Dumps
 - user abend code 319
- DUMPSERV address space 143
- DUPDTS statistic
 - description and offset 553
- dynamic dispatching 141
- dynamic file allocation
 - bypassed, if not necessary 382
 - using TIOT and XTiot 497
- dynamically allocated datasets
 - how recovery works 382
 - recovering 382

E

- Early Warnings
 - installation information 282
- EBM pages
 - and above the bar storage 41
- ECCALL statistic
 - description and offset 553
- ECCNCT statistic
 - description and offset 553

- ECCTOUT statistic
 - description and offset 553
- ECCWAITM statistic
 - description and offset 553
- ECCWAITS statistic
 - description and offset 554
- ECDELETE statistic
 - description and offset 554
- ECF
 - \$STATUS return codes 526
 - \$STATUSD return codes 526
 - images 520
 - overhead loading programs 529
 - restrictions and cautions 530
 - subtask assignment 529
 - subtask management 529
 - User 0 parameters 528
 - See also* External Call Facility
- ECISUBS parameter
 - setting for subtask affinity 530
- ECKD channel program support 170
- ECLOAD statistic
 - description and offset 554
- ECMODMAX statistic
 - description and offset 554
- ECMSUBS parameter
 - setting for subtask affinity 530
- ECNAMMAX statistic
 - description and offset 554
- ECTSKMAX statistic
 - description and offset 554
- ECTWAITM statistic
 - description and offset 554
- ECTWAITS statistic
 - description and offset 554
- EDIT user parameter 106
- ENABLE SUBSYSTEM FILE command 184
- encryption 249
- ENQCTL command 35
- Enqueuing
 - and logical inconsistencies 363
- ENTER macro 239
- EOF markers 393
- EOF option
 - and reading CCAJRN dataset 395
 - UTILJ utility 394
- EOJ command
 - terminating XDM 81
- EOJ,CANCEL command
 - usage warning 82
- EOJ,FORCE command
 - usage warning 82
- ER message type 321
- ERMx statistic 36

- ERRMSGL parameter
 - saved error messages 292
- ERROR clauses
 - of RESTART command 358
- error conditions
 - MERGEJ utility 404
- error diagnostics
 - for IFAM 1 481
- error handling
 - ERRMSGL parameter 292
- error messages
 - changing type 321
 - SUBSYSMGMT 197
 - general 197
- Error processing
 - SUBSYSMGMT 188
 - subsystems 188
- errors
 - correcting recovery errors 377
 - determining the cause of recovery failure 377
- ERRPDL statistic
 - description and offset 554
- ESPIE macro routine 337
- ESTAE exit macro 336
- ESTAE macro routine 337
- example code
 - BUMP command 521
 - JCL for z/OS ZCTLTAB utility 263
 - JCL for z/VSE ZCTLTAB utility 264
- example programs
 - a wait for an extended quiesce 390
 - third-party backups 389
- EXCPVR feature
 - enabling and using 434
 - relationship to IOS BRANCH ENTRY 434
 - saving CPU time 433
- EXEC statement 23
 - parameters and IFAM1 481
- EXECs
 - AUDIT204 317
- extended quiesce
 - and ring stream journals 389
 - definition of 388
 - ending 393
 - limit on \$POST calls 392
 - managing 392
 - managing the duration of 389
 - programming a wait for 390
 - status messages 392
 - understanding 388
- Extended Task Input/Output Table. *See* XTIO 497
- extension records
 - overflow pages 355
- external backup

- coordinating 389
- External Call Facility
 - definition of 519
 - loading a program 520
 - See also* ECF
- EXTERNAL CALL statement
 - external module availability 530
 - invoking an external module 530
 - syntax for 522
- EXTERNAL DELETE statement
 - syntax for 523
- EXTERNAL LOAD statement
 - reloading a previously-loaded module 530
 - syntax for 523
- external modules
 - reloading 530
- EXTERNAL NAME statement
 - syntax for 525
- EXTERNAL PROGRAM statement
 - syntax for 523
- External programs
 - acceptable work 520
 - loading 523
 - logical name 520
 - subtask assignment 520
- external programs
 - stopping 521
- EXTERNAL START statement
 - syntax for 525
- EXTERNAL statements
 - return codes 526
- EXTERNAL STOP statement
 - syntax for 526
 - with FORCE option 521

F

- Familiar CRAM. *See* CRAM
- fast read
 - definition of 461
 - DKPRF statistic 461
 - tracking with DKPRF statistic 461
- fast reads
 - improving with MAXOBUF setting 461
 - improving with SCHDOPT setting 461
- FBA devices 168
- FBWT statistic
 - description and offset 554
- FDIR option
 - UTILJ utility 394
- Fields
 - security table 57
- file discontinuity

- definition of 362
- file groups
 - ad hoc 157
 - CCATEMP, use of 147
 - creation of 113, 159
 - definitions 161
 - deleting 161
 - enqueueing 161
 - FTBL storage 51
 - permanent 156
 - sharing of 161
 - temporary 156
 - updating 158
 - uses for 157
- file processing facility 410
- FILENAME option
 - UTILJ utility 394
- files
 - discontinuities 362
 - integrity 117
 - multiple names 158
 - number that can be regenerated 385
 - opening and RESTART recovery status 369
 - reasons not recovered by ROLL BACK processing 358
 - report after RESTART recovery 368
 - roll-forward-all-the-way 361
 - statistics 586, 587
 - final 586
 - partial 587
 - transaction back out 361
 - types used in ROLL FORWARD processing 361
 - VSAM, loading of 517
- files, allocation of 492
- final statistics
 - file 586
 - system 563
 - user 575
- FINDS statistic
 - description and offset 554
- FISTAT parameter
 - RESTART recovery status 369
- fixed table size 47
- FLOD
 - exit program 147
 - NTBL usage 54
 - QTBL usage 54
 - STBL usage 58
 - VTBL usage 61
- FLOD exit
 - CCATEMP space 147
- FLOD exits
 - space requirements 24
- FOPT parameter 356
 - transaction back out 356
- FORCE option
 - when activated 522
- FORMAT command
 - syntax for 309
- FORMAT option
 - UTILJ utility 394
- FORTRAN application language
 - with Model 204 HLI 479
- FRCVOPT parameter 33, 360, 361
 - and shared files 33
 - Checkpoint facility 333
 - disabling transaction back out 356
 - for ROLL FORWARD processing 362
 - journal generation 292
 - ROLL FORWARD processing 360
 - roll-forward-all-the-way files 361
 - transaction back out files 361
- FREE command 500
- FROMDATE option
 - behavior without times options 396
 - behavior without TODATE option 396
 - UTILJ utility 394
- FROMTIME option
 - behavior without dates options 396
 - behavior without TOTIME option 396
 - UTILJ utility 394
- From-to date-time options
 - UTILJ utility 396
- FSATTN user parameter 106
- FSCB statistic
 - description and offset 554
- FSCBSW statistic
 - description and offset 554
- FSTRMOPT user parameter 106
- FTBL 51
 - file group storage 51
- FTBL statistic
 - description and offset 554
- Full screen buffer table (FSCB)
 - storing definitions 50
- functions
 - \$CODE 241
- functions, user-written
 - coding of 237
 - LEAVEF0 macro 240
 - LEAVENUM macro 240
 - LEAVESTR macro 240
 - NOARG option 239
 - procedures 236
 - return value, numeric 240
 - return value, string 240
- FUNU module 236

G

- GDG option
 - using, with cautionary note 420
- GDG streams
 - definition of 419
- GDGs
 - and Dataset Control Blocks 420
- GETVIS area
 - z/VSE partition 481
- GETVIS area (z/VSE) 10
- Global Variable Table (GTBL)
 - sizing 181
 - understanding 52
- global variables 52
- Groups
 - security table 57
- groups, See file groups
- GTBL 52
 - minimum length 53
- GTBL sizing 181
- GTBL statistic
 - description and offset 554
- GTBLRS statistic
 - description and offset 554
- GTBLRU statistic
 - description and syntax 554

H

- HALT command 144
- hash cells
 - buffer pool data structure 38
- HASHCELL parameter
 - allocation hash cells 42
 - Model 204 parameters
 - HASHCELL 38
 - NUMBUG 43
- HDR1 parameter 502, 509
- HDR2 parameter 502, 509
- HDR3 parameter 509
- HDRCTL user parameter 106
- HEAP statistic 454
 - description and offset 554
- histograms
 - UTILJ report 396
- HLI output
 - LOBUFF parameter 483
- Horizon 96
 - description of 11
- Host Language Interface
 - CRAM requirements 92
 - CSA storage 84, 85

- IFCMMT call 352
- IFSTRT protocol 92
 - transaction back out 352
- HTLEN user parameter 106

I

- IBM documentation
 - and SVCs 592
- IFAM
 - CCATEMP file 147
 - FTBL storage 51
 - IFUTBL function 46
 - NTBL entries 54
 - QTBL usage 54
 - STBL usage 58
 - system scratch file 147
 - VTBL usage 61
- IFAM 1
 - required label information 482
- IFAM1
 - and z/OS 480
 - cataloging programs 482
 - CCATEMP file (DOS) 153
 - configuration 5
 - DD statement required for error diagnostics 481
 - definition of 480
 - EXEC statement parameters 481
 - file locking 33
 - improving performance with Model 204 480
 - loaded dynamically with Model 204 480
 - region where run 480
 - required DD statements 480
 - system scratch file 153
- IFAM2
 - configuration 5
 - CRAM requirements 92
 - customized with IFIF 484
 - definition of 482
 - file locking 33
 - IFAMFHALT command 114
 - IFAMFORCE command 114
 - IFAMOPEN command 114
 - IFAMSTART command 114
 - IFAMSTAT command 114
 - IFSTRT protocol 92
 - IODEV thread 100
 - performance 452
 - z/OS operating system 92
 - z/VSE operating system 92
- IFAM4
 - *SLEEP command 112
 - configuration 5

- file locking 33
- SPCORE requirements 29
- IFAMBS parameter 85
 - calculating 483
 - definition of 483
- IFAMCHNL parameter 83
- IFAMFORCE command 114
- IFAMHALT command 114
- IFAMOPEN command 114
- IFAMPROD channel name 93
- IFAMSTART command 114
- IFAMSTAT command 114
- IFCHKPNT 333
- IFCMMT call
 - committing data in HLI programs 352
 - lock-pending-updates 354
- IFDIAL connection
 - in ONLINE command 19
- IFDIAL protocol 93
- IFDIAL threads
 - support for 484
- IFFNSH 333, 335
- IFIF interface module
 - customized IFAM2 484
- IFPPCI module 542
- IFRECV command 542
- IFRECV function 542
- IFSEND function 542
- IFSENDX command 542
- IFSETUP function 20
- IFSGNL command 542
- IFSGNL function 542
- IFSTRT 333, 335
- IFSTRT protocol 92
- IFUTBL function 46
- IICB
 - sizing CRAM channel storage 90
- Image definitions
 - stored in FSCB 50
- images
 - stored above the bar 42
- implementation considerations 339
- IN statistic
 - description and offset 554
- In Storage feature
 - APSY precompiled procedures 178
 - CCASERV 151, 166
 - CCATEMP 151, 166
- INBUFSIZE parameter
 - CRAM SQL 455
 - Horizon SQL 455
- INCCC parameter 84
- INCCC user parameter 106
- INCMFS statistic
 - description and offset 554
- INCMIO statistic
 - description and offset 554
- inconsistencies
 - resolving logical 363
- inconsistency, file 363
- inconsistent files
 - REGENERATE command 386
 - repairing 364
- INCRAM statistic
 - description and offset 554
- in-house statistical processing
 - and 64-bit processing 9
- initialization
 - subsystems available 177
- INITIALIZE command 274
- INMRL parameter 84, 95
- INMRL user parameter 106
- INPUT user parameter 106
- INTERCOMM interface 94
- Internal Interregional Control Block. See IICB
- Internal statement table
 - QTBL 54
- Inverted Files Access Methods
 - interfaces for 479
- INVMFS statistic
 - description and offset 554
- INVMIF statistic
 - description and offset 555
- INVMIO statistic
 - description and offset 555
- INXX statistic
 - description and offset 555
- IODEV parameters 64, 99
 - IODEV=39 99
- IODEV threads, SQL
 - defining 94, 96
- IODEV user parameter 106
- IODEV=11 74
- IODEV=19 94
- IODEV=23 74, 92
- IODEV=27 96
- IODEV=29 74, 84
- IODEV=29 statement 84
 - required parameters 84
- IODEV=3 67
- IODEV=37
 - 3767 terminals 72
- IODEV=39 98, 99
- IODEV=41 98, 100
- IODEV=43 98, 100
- IODEV=7
 - 3270 terminals 72
- IOS BRANCH ENTRY

- relationship to EXCPVR 434
- IOS Branch Entry
 - and XTIO usage 498
 - saving storage 435
- IOS BRANCH ENTRY feature
 - description 434
- IOSLICE scheduling parameter 141
- IPADDR parameter
 - retrieving an IP address 73
- ITBL 53
- ITBL statistic
 - description and offset 555
- IUCV 97
- IUCV full-screen thread 100
- IXADD statistic
 - description and offset 555
- IXDEL statistic
 - description and offset 555

J

- JCL
 - AUDIT204 (z/OS) 313, 316
 - CCAJRNL requirements 287
 - CCASNAP file 324
 - CCATEMP (DOS) 152
 - CCATEMP (z/OS) 150
 - for using CCASYS 177
 - reusing for RESTART recovery 377
 - ring/parallel journal stream 423
 - UTILC utility 344
 - VSE
 - ring/parallel journal stream 423
 - UTILC 345
 - z/OS 345, 423, 425, 427
 - concatenated offload stream 427
 - parallel checkpoint stream 425
 - UTILC utility 345
 - z/VSE 345, 423, 425
 - parallel checkpoint stream 425
- JCL for a recovery run
 - z/OS 372
 - z/VSE operating system 372
- JDBC for Model 204
 - description of product 11
- JES log
 - M204CKPX exit 336
- job step return codes, See return codes
- journal
 - file processing facility 410
 - ring/parallel stream (z/OS) 423
 - streams 411, 416, 417, 423, 424
 - concatenated 416

- parallel 417
- ring 411
- ring/parallel stream (CMS) 424
- ring/parallel stream (z/VSE) 423
- journal datasets
 - and ROLL FORWARD processing 361
 - EOF markers 393
 - in ROLL FORWARD logging 362
- journal files
 - crossing versions 287
 - merging overlapping 401
 - using the correct journal 297
- journal processing
 - media recovery 383
- journaling
 - without interruption 300
- journals
 - buffer size 300
 - for multiple channel programs 288
 - in VM/CMS 294
 - JCL requirements 287
 - NJBUFF parameter 300
 - storing update records 287
- JRIO module 360

L

- label information
 - required for IFAM1 482
- large datasets
 - support for 494
- LAUDIT parameter 106
 - values of 300
- LCPDLST user parameter 106
- LEAVENUM macro 240
- LEAVESTR macro 240
- LECHO user parameter 106
- LENQTBL parameter 27
 - specifies 27
- LFSCB parameter 106
- LFSCB user parameter 106
- LFTBL parameter 106
- LFTBL user parameter 106
- LGTBL parameter 106, 181
 - and Parallel Query Options/204 181
- LGTBL user parameter 106
- LIBUFF parameter 15, 27, 48, 106
 - length of HLI functions 483
- LIBUFF user parameter 107
- Line-at-a-time protocol 93
- line-at-a-time thread 99
- LITBL parameter 107
- LKPOST statistic

- description and offset 555
- LKPOST statistics 461
- LKWAIT statistic
 - description and offset 555
- LKWAIT statistics 461
- LM 243
- LNTBL user parameter 107
- Load Multiple 243
- LOBUFF parameter 27, 48
 - HLI output 483
- LOBUFF user parameter 107
- locking conflicts 35
 - handling 33
- lock-pending updates
 - locking mechanism 354
- lock-pending-updates
 - with ROLL FORWARD processing 361
- LOGADD parameter 27, 255, 256
- LOGCTL command 114, 254, 256, 277
 - file-level security 271
 - modifying password table 265
 - terminal-level security 277
- LOGFAIL parameter 27, 252, 258
- LOGFILE command 272
 - description 115
 - terminal-level security 277
- logging
 - ROLL FORWARD processing 365
- logging out users 113
- LOGGRP command 273
 - description 115
 - terminal-level security 277
- logical record length
 - sequential processing 513
- logically inconsistent files
 - resolving 363
- Login delays
 - and LOGWHO command 253
 - automatic security feature 253
 - invoking security 253
 - STATUS command 253
- login entries 254, 277
 - LOGCTL command 254, 271
- Login failure
 - MONITOR command 253
- login processing
 - SUBSYSMGMT 185
- login, automatic 204
- LOGKEY command 115
- LOGLST command 254
 - description 115
 - displaying user ID password tracking 258
 - terminal-level security 277
- LOGONENQ parameter 27

- LOGONENQ user parameter 107
- logout, automatic 204
- logs
 - constraints 365
 - transaction back out 365
- LOGTRY parameter 27, 252, 258
- LOGWHO command
 - unsuccessful logins 253
- LONGUPDIME statistic
 - description and offset 555
- LONGUPDTS statistic
 - description and offset 555
- Look-ahead read, See prefetch feature
- LOUTPB parameter 85
 - in pseudo conversational CICS 543
- LOUTPB user parameter 107
- LPDLST user parameter 107
- LPU. See lock-pending-updates
- LQTBL user parameter 107
- LRETBL 32
- LRETBL parameter 27
- LRUTIM parameter 27
 - specifies 27
- LSERVPD parameter 166
- LSTBL user parameter 107
- LTTBL user parameter 107
- LU 6.2 sessions 97
- LVLTRC user parameter 108
- LVTBL user parameter 108
- LXTBL user parameter 108

M

- M204 command 101
- M204APND utility 21
- M204CKPX ASSEMBLER exit
 - checkpoint user exit 336
- M204CKPX user exit 335
- M204CMS utility 21
- M204FDEF utility 21
- M204FULL channel name 92
- M204LDEF utility 21
- M204MOUN utility 22
- M204PROD channel name 93
- M204UNLD EXEC 22
- M204USR LOADLIB file 543
- M204USR MODULE 99
- M204UTIL 21
- M204XFER utility 21
- M204XSVC 457
- macros
 - CODETABL 241
 - ENDTAB 242

- LEAVEF0 240
- LEAVENUM 240
- LEAVESTR 240
- PGFIX 440
- maintaining journal files
 - for media recovery 349
- managing
 - checkpoint extended quiesce 389
 - third-party backups 388
- mandatory member 182
- Mandatory members
 - disabling 185
- manual member 182
- Master/slave mode
 - CRAM 83
- MAXBUF parameter 27, 38
- MAXHDR user parameter 108
- MAXOBUF parameter
 - improving fast page reads 461
- MAXSIMIO parameter
 - allocating DBIDs 435
- MAXSPINS parameter 461
- MAXTRL user parameter 108
- Mean statistical measure
 - definition of 308
- media recovery 383, 388
 - NDCBS parameter 385
 - NDIR parameter 385
 - NFILES parameter 385
 - phases of 383
 - REGENERATE command 385
 - required datasets 386
- media recovery run
 - for z/OS operating systems 387
- members, subsystem 182
- MEMLIMIT
 - real storage above 2G address 8
- MEMLIMIT system option
 - setting 8
- Menu definitions
 - stored in FSCB 50
- MERGEJ utility 350, 401
 - CMS operating system code 406
 - datasets required 404
 - error conditions 404
 - output entries 591
 - using 401
 - z/OS JCL 405
 - z/VSE operating system code 406
 - z/VSE operating system code example 405
- message display
 - BROADCAST command 113
- message type
 - modifying 321

- messages
 - displayed for ROLL BACK processing 379
 - error in refreshing subsystem procedures 229
 - MSG command 115
 - MSGCTL command 115
 - successfully refreshing subsystem procedures 228
- Migrated datasets
 - recalling 71
- MINAVAIL parameter 418
- MINBUF parameter 27, 38
- Model 204
 - configurations 5
 - description 4
 - file structures 4
 - hardware requirements 5
 - related products described 11
- Model 204 commands
 - ALLOCATE 498
 - CHKABORT 335
 - DEFINE 498
 - DESECURE 272
 - DUMPG 161
 - in precompilable procedures 234
 - LOGCTL 255
 - LOGFILE 277
 - LOGGRP 277
 - LOGLST 258, 277
 - obsolete form of ALLOCATE 498
 - REGENERATE IGNORE 384
 - SECURE 272
 - STATUS 369
 - SWITCH STREAM 288, 290
 - TMASKUPDATE 277
- Model 204 image
 - external programs 520
- Model 204 maintenance applied
 - keeping track of 282
- Model 204 pages 4
- Model 204 parameters
 - CDMAXP2X 356
 - CDMINP2X 356
 - DSOPT 179
 - ERRMSG 292
 - HASHCELL 42
 - IPADDR 73
 - LIBUFF 483
 - LOBUFF 483
 - LOGFAIL 258
 - LOGTRY 258
 - NUMBUF 38
 - NUMBUFG 40
 - PWDEXP 258
 - PWDPURGE 258

- PWDWARN 258
- RCVOPT 380
- SYSOPT2 498
- TEMPPAGE 179
- Model 204, Release 9.0
 - REGENERATE command 386
- Model 294 commands
 - ANALYZE 308
- MODEL user parameter 108
- MODEL204 MODULE file 543
- modules
 - CDTB
 - description 241
 - sizing of 242
 - FUNU 236
 - IFPPCI 542
- MONITOR ACTIVE command
 - usage 125
- MONITOR command 120, 133
 - BASIC 126
 - description 115
 - DISKBUFF 126
 - ENQ 126
 - file listing 120
 - for disk buffers 9, 43
 - for XDM 79
 - LINK 126
 - PERFORMANCE 125
 - performance statistics 586
 - PROCESS 130
 - PROCESSGROUP 128
 - pseudo subtasks 133
 - STATISTICS 125
 - unsuccessful logins 253
 - USERS ALL 125
 - VTAM 126
- MONITOR CONFLICT command 584
- MONITOR DISKBUFF commands
 - using 131
- MONITOR SUBSYSTEM command 458
- MONITOR TASKS command 461
- Monitoring
 - VTAM interface 120
- monitoring 231
 - basic system information 120
 - dynamic storage allocation 449
 - formatted displays 120
 - unformatted displays 120
 - user information 120
 - VIEW command 116
- MOVE statistic
 - description and offset 555
- MP lock
 - spinning on for improved performance 463

- MP/204 458
 - description of product 11
 - fast logical page reads 461
 - subtask statistics 573
 - user-written functions 245
- MQ/204
 - description of product 12
- MQAPICNT statistic
 - description and offset 555
- MQAPITIM statistic
 - description and offset 555
- MQBYTEIN statistic
 - description and offset 556
- MQBYTEOU statistic
 - description and offset 556
- MQGETS statistic
 - description and offset 556
- MQGWCNT statistic
 - description and offset 556
- MQGWTSUC statistic
 - description and offset 556
- MQGWTTIM statistic
 - description and offset 556
- MQGWTTSP statistic
 - description and offset 556
- MQHWQU statistic
 - description and offset 556
- MQHWTASK statistic
 - description and offset 556
- MQNUMQM statistic
 - description and offset 556
- MQNUMQU statistic
 - description and offset 556
- MQPUTS statistic
 - description and offset 556
- MQWTM statistic
 - description and offset 556
- MS message type 321
- MSG command 115
- MSGCTL command 115, 368
 - assigning independent options 323
 - job step return codes 134
 - manipulating message type 320
 - NOACTION option 322
 - options 320
 - use 299
- multiple file names 158
- multiple jobs 33
 - CCATEMP file 149
- multiple Model 204 versions 34
- multiple procedure files 208
- multiprocessing 245, 458

N

- NBKPG user parameter 108
- NDCBS parameter 28
 - and CCAGRP 161
 - media recovery 385
- NDIR parameter 28
 - and CCAGRP 161
 - media recovery 385
 - ROLL BACK processing 358
- New features
 - ECF for z/OS 519
- NFILES
 - and CCAGRP 161
- NFILES parameter 28
 - media recovery 385
 - ROLL BACK processing 358
- NGROUP parameter 28
- NJBUFF parameter
 - journal buffer size 300
- NLBUFF parameter
 - setting buffers for CCAJLOG 297
- NOACTION option
 - MSGCTL command 322
 - processing 322
- NOARG option 239
- NOAUDITxx options
 - processing 323
- NOCOUNT option
 - caution 321
- non XA systems 38
- NonStop/204
 - using third-party backups 388
- NORQS user parameter 108
- Notation conventions xxv
- NOTE/POINT facility
 - handling CCARF datasets 381
- NOTERM parameter 97
- NOTERM user parameter 108
- NOTHREAD parameter 94, 97
- NOUTBUF user parameter 108
- NSERVS parameter 28, 169
- NSUBTKS parameter 28, 95, 334
 - Checkpoint facility 333
 - statistics, setting for 302
- NSUBTKS user parameter 108
- NTBL 456
- NTBL statistic
 - description and offset 556
- NUMBUF 38
- NUMBUF parameter
 - disk buffers allocated 38
- NUMBUFG parameter
 - allocating above the bar buffers 40

- determining the setting 41
- HASHCELL 43

- NUMLK 208
- NUSERS parameter 28, 169
 - and sub-transaction checkpoints 339

O

- Off loading
 - AUTOOFFLOAD option 413
 - COPY command 414
 - OFFLOAD command 413
 - ring stream 411, 413
- OFFIN statistic
 - description and offset 556
- OFFLOAD command 115
- OFFOU statistic
 - description and offset 556
- ONLINE 134
 - CCATEMP file 153
 - MONITOR command 115
 - performance monitoring 120
 - sample z/OS data stream 16
 - status monitoring 120
 - system scratch file 153
 - termination of 117
 - *SLEEP command 118
 - automatic 118
 - by operator 118
 - by system manager 117
- ONLINE command 19
- ONLINE configuration 5
- OPENCTL command 270
 - file classification 270
 - record security 274
- operating systems
 - recovery dataset required 371
- Operational Parameters screen
 - changing attributes 227
- optimizing performance
 - managing the constraints log 355
- optional member 183
- Optional members
 - disabling 185
- OUT statistic
 - description and offset 556
- OUTCCC parameter 84
- OUTCCC user parameter 109
- OUTCMFS statistic
 - description and offset 556
- OUTCMIO statistic
 - description and offset 556
- OUTCRAM statistic

- description and offset 556
- OUTLPP user parameter 109
- OUTMRL parameter 84
- OUTMRL user parameter 109
- OUTPB statistic
 - description and offset 557
- OUTPUT user parameter 109
- output, See directed output
- OUTVMFS statistic
 - description and offset 557
- OUTVMIF statistic
 - description and offset 557
- OUTVMIO statistic
 - description and offset 557
- OUTXX statistic
 - description and offset 557
- overflow pages
 - extension records 355

P

- page reads
 - fast logical 461
- page release 456
- PAGEFIX parameter 28
 - use 440
- pages 4
- parallel checkpoint stream
 - example 424
- Parallel Query Option/204 147
 - description of product 12
 - FTBL requirements 52
 - resetting LGTBL parameter 181
 - scattered subsystems 181
 - STBL requirements 58
- parallel stream
 - description 417
 - error processing (input) 419
 - error processing (output) 418
- parallel streams
 - limits to 417
 - switch processing 291
 - SWITCH STREAM command 290
- Parameters
 - FRCVOPT
 - journal generation 292
 - LOGADD 255
 - NJBUFF 300
 - NSUBTKS 302
 - RCVOPT
 - journal generation 292
 - UPDTID 368
- parameters
 - ACCTIM 576
 - AUTOSYS 191, 196
 - CPMAX 342, 411
 - restrictions 411
 - HDR1 502, 509
 - HDR2 502, 509
 - HDR3 509
 - LOGFAIL 252
 - LOGTRY 252
 - MINAVAIL 418
 - PAGEFIX 440
 - PRIVDEF 270
 - PROCFIL 208
 - SCOPE 502
 - SEP 502, 509
 - SMFLORN 593
 - SMFSLRN 593
 - SMFSVC 593
 - SYSOPT 177, 252, 299, 305
 - CCAJRNL 299
 - SYSOPT set for an audit trail 305
 - TERMID 502
 - TRANSID 502
 - UDDRFM 515
- parameters, See scheduler parameters
- PARM parameter 24
- partial statistics
 - file 587
 - system 563
 - user 576
- partner process 542, 545
- password 273
- Password Expiration feature
 - at your site 265
 - deleting CCASTAT entries 259
 - enhancing CCASTAT security 258
 - testing at your site 260
 - updating CCASTAT 260
 - ZCTLTAB utility 260
- Password table
 - login entries
 - addition of 255
 - LOGADD parameter 255
 - LOGCTL command 255
 - updates, processing of 256
 - updates, storage allocation 256
 - login privileges 253
 - login security 253
- password table
 - adding file entries 271
 - changing terminal entries 276
 - deleting terminal entries 276
 - field level entries
 - addition of 274

- changing of 274
 - deletion of 274
- field level security entries
 - addition of 276
 - changing of 276
 - deletion of 276
- file entries 271
 - index 273
 - listing of 272
 - LOGFILE command 272
- group entries
 - addition of 273
 - deletion of 273
 - listing of 273
- group entries classifications of 273
- LOGFILE command 115
- login entries
 - backup 255
 - change of 255
 - deletion of 255
 - listing of 254
 - LOGCTL command 255
 - LOGLST command 254
- login entries in z/VSE 251
- LOGKEY command 115
- LOGLST command 115
- storage format 254
- updating in z/VSE 251
- passwords
 - changing and reusing 266
 - defining 266
 - defining for Password Expiration feature 259
 - FILE and GROUP 259
 - from a trusted environment 267
 - revoking 258
 - tracking validity 258
 - waiting to change 266
- Pattern matcher
 - using STBL 59
- PBRNFLT statistic 454
 - description and offset 557
- PCPU statistic
 - description and offset 557
- PDL statistic 454
 - description and offset 557
- Performance
 - record-number retrieval 451
 - statistics 563
 - system 563
- performance
 - channel load balancing 448
 - conserving core 449
 - CRAM 448
 - and IFAM2 448
 - and User Language 448
 - I/O z/OS, z/OS/XA) 434
 - IFAM2 114, 452
 - IFAMHALT command 114
 - IOS BRANCH ENTRY 434
 - operating system recovery 455
 - page fixing 440
 - PAGEFIX parameter 440
 - PGFIX macro 440
 - server swapping 448
 - statistics 585
 - user 585
 - Timer SVC 433
- performance enhancements
 - dynamic file allocation 382
 - finding checkpoints quickly 381
 - spinning on an MP lock 463
- Performance improvements
 - using XDM 76
- PGFIX macro 440
- PGM parameter 24
- PGRLSE 456
- PL/I application language
 - with Model 204 HLI 479
- PNDGTIME statistic
 - description and offset 557
- POLLNO user parameter 109
- posting
 - CPQZ 389
 - QZSIG 390
- PQO 147
- PR statistic
 - description and offset 557
- precompilable procedures
 - processing for subsystems 230
 - using with commands 234
 - with commands 234
 - with dummy strings 234
- Prefetch feature
 - performance gains 451
- preimages
 - definition of 351
 - ROLL BACK facility 357
- primary pages
 - constraints database 355
- PRIOMAX scheduling parameter 141
- PRIORITY command 116
- priority scheduling 116
 - dynamic dispatching 141
 - HALT command 144
 - queue aging 143
 - User 0 144
- PRIORITY user parameter 109
- PRIVDEF parameter 270

- Privilege byte, values for 253
- problem identification
 - CCASNAP file 320
 - MSGCTL command 320
 - SNAPCTL parameter 320
- procedure compilation
 - dynamic refresh 228
- procedure dictionary 208
- procedure files 208
- procedure security table (XTBL) 61
- processing statistics
 - third-party applications 9
- PROCFIL 208
- PROCFIL parameter 208
- Program Communication facilities
 - CICS
 - IFPPCI module 542, 543
 - IFRECV function 542
 - IFSEND function 542
 - IFSGNL function 542
 - pseudo conversational 543
 - CMS 543
 - IFRECV command 542
 - IFSENDX command 542
 - IFSGNL command 542
 - M204USR LOADLIB file 543
 - partner process 542, 545
- Program Communication facility
 - CMS
 - MODEL204 MODULE file 543
- program-to-program processing 96, 542
- pseudo subtasks
 - CHKAWW 334
 - CHKPPST 134
 - CHKPTIMO 134
 - CHKPTIMR 134
 - description of 132
 - for statistics 302
 - MONITOR command 115
 - offloading 411
 - space requirement 132
 - statistics 133
- PWDEXP parameter
 - days of use for a valid password 258
- PWDPURGE parameter
 - days a user ID is suspended 258
- PWDWARN parameter
 - days prior to password expiration 258

Q

- QTBL 456
 - definition and description of 54

- QTBL statistic
 - description and offset 557
- queue aging 143
- QZSIG
 - posting and unposting 390

R

- RCVOPT parameter 360
 - Checkpoint facility 333
 - for ROLL FORWARD processing 362
 - journal generation 292
 - setting for RESTART recovery 380
 - setting for ROLL FORWARD processing 360
- REACTIVATE command 116
- RECADD statistic
 - description and offset 557
- RECDEL statistic
 - description and offset 557
- RECDS statistic 454
 - description and offset 557
- record format
 - sequential processing 512
- record types, checkpoint 343
- Record-number retrieval
 - Prefetch feature 451
- records
 - backing out 365
- Recoverable errors
 - subsystems 188
- Recovery
 - file types 361
- recovery 113
 - audit trail messages 368
 - automated 377
 - CCASYS file 178
 - CCATEMP file 147
 - checkpoint stream (z/VSE) 425
 - CHKABORT command 113
 - DEFINE STREAM (offload) 426
 - DEFINE STREAM (recovery) 422
 - determining the cause of failure 377
 - errors 377
 - files 332, 410
 - processing of 410
 - files and checkpoints 332
 - flags 561
 - values of 561
 - for dynamically allocated datasets 382
 - forcing a successful rerun 378
 - handling failures 377
 - media 388, 400
 - operating system performance 455

- parallel checkpoint stream (CMS) 426
- parallel stream (z/OS) 425
- REGENERATE command 116
- RESTART command 116
- RESTOREG command 116
- ring stream input 415
- ring/parallel stream 421
- ring/parallel stream (CMS) 424
- ring/parallel stream (z/OS) 423
- ring/parallel stream (z/VSE) 423
- ROLL BACK facility
 - ROLL BACK facility
 - how it works 357
- ROLL FORWARD facility
 - ROLL FORWARD facility
 - RESTART recovery 359
- START command 116
- statistics 562
 - entry types 562
- stream configurations 410, 411, 416, 417
 - concatenated 416
 - parallel 417
 - ring 411
 - system 421
 - tracking updates 379
 - user hard restart 367
 - z/OS concatenated stream 427
- recovery datasets
 - required by operating systems 371
- RECTYPE option
 - UTILJ utility 395
- recursions
 - limits to 417
- REDY statistic
 - description and offset 557
- REFRESH SUBSYSTEM command
 - privileges 229
- REGEN processing
 - with one-at-a-time CCAGEN 384
- REGENERATE command 116, 351, 383
 - BEFORE clause 383
 - IGNORE option 384
 - inconsistent files 386
 - media recovery 385
 - Model 204 Release 9.0 386
- REGENERATE processing
 - and checkpoint quiesce 290
- REGION parameter 24
- Region size
 - calculating 24
- region size
 - using statistics 449
- RELEASE 354
- Remote User Language threads 75
- reorganizing CCASYS 177
- REPORT command
 - syntax for 311
- REPORT option
 - UTILJ utility 395
- reports
 - on ROLL FORWARD processing 368
- REQ statistic
 - description and offset 557
- required datasets
 - media recovery 386
 - MERGEJ utility 404
- requirements
 - for BATCH2 backup 392
 - storage for APSY precompiled procedures 179
- RESCURR parameter 457
- RESERVE/RELEASE 34
- RESHIGH parameter 457
- resident requests 456
- resource locking 31
- resource locking table size 33
- RESSIZE parameter 457
- REST statistic
 - description and offset 557
- restart
 - z/VM procedure name 19
- RESTART command 116, 351
 - ERROR clauses 358
 - fixing files 367
 - invoking RESTART recovery 351
 - status report values 370
 - with deferred update datasets 382
- RESTART recovery
 - and sub-transaction checkpoints 366
 - bypassing dynamic file allocation 382
 - file status reported 368
 - FISTAT parameter 369
 - NDIR parameter 381
 - NFILES parameter 381
 - rerunning after successful recovery 378
 - suppressing status messages 369
 - using TIOT or XTiot 497
- RESTHRSH parameter 457
- RESTOREG command 116, 161
- retrieving an IP address IPADDR 73
- RETRVKEY parameter 28
- RETRVKEY user parameter 109
- RETRYA statistic
 - description and offset 557
- RETRYC statistic
 - description and offset 557
- return codes
 - job step return codes 134

- UTILJ utility 398
- ring configuration
 - as input 415
 - EOD processing 119
 - off loading 119
 - OFFLOAD command 116
- ring stream 414
 - CPMAX parameter 411
 - description 411
 - off loading 413
- ring stream journals
 - and extended quiesce 389
- ring/parallel journal stream FILDEF 424
- RK message type 321
- Rocket Software products
 - related to Model 204 11
- Rocket Software Technical Support
 - evaluating your CCAPRINT and CCAAUDIT files 282
- ROLL BACK facility
 - and ROLL FORWARD processing 364
 - preimages 357
- ROLL BACK processing
 - CHKPOINT dataset required 379
 - forcing 380
 - how it works 357
 - messages displayed 379
 - NDIR parameter 358
 - NFILES parameter 358
 - Phase 1 described 357
 - reasons files were not recovered 358
 - sub-transaction checkpoints 366
- ROLL FORWARD facility
 - calculating SPCORE 367
 - for deferred update files 382
- Roll Forward facility 359
 - journal parameters 292
- ROLL FORWARD processing
 - and sub-transaction checkpoints 366
 - back outs during 365
 - CCARF dataset 379
 - estimating time to complete 379
 - file types 361
 - logging 365
 - operational changes 379
 - reporting on 368
 - run separately 379
 - steps involved 360
 - user hard restarts 364
- roll- forward-all-the-way files
 - FRCVOPT setting 361
- RQTM statistic
 - description and offset 558
- RSXCOMP statistic

- description and offset 558
- RTBL 57
- RUNG statistic
 - description and offset 558
- runtime
 - environment specifications 23
 - User 0 parameters 26
- Runtime actions
 - SYSPOPT parameter 25
- runtime parameters
 - CFRLOOK 27
 - CPMAX 27
 - CPTIME 27
 - LENQTBL 27
 - LRUTIM 27

S

- sample programs
 - submitting a backup 391
- SAMPING EXEC 20
- save compilation pages
 - without duplication 179
- scattered subsystems
 - Parallel Query Option/204 181
- SCHDCPU statistic
 - description and offset 558
- SCHDOPT parameter 245
 - improving fast page reads 461
- Scheduler parameters
 - listing of 141
- scheduler parameters
 - viewing of 141
- SCOPE parameter 502
- scratch file, See CCATEMP
- Screen definitions
 - stored in FSCB 50
- screens
 - stored above the bar 42
- SCREENS statistic
 - description and offset 558
- secondary recovery
 - automated 377
- SECURE command 272
- Security
 - login delays 253
 - login password table 253
 - monitoring login 253
 - RTBL 57
- security
 - adding file-level 272
 - adding file-level entries 271
 - CCASYS file 177

- CREATE GROUP command 273
- deleting file-level entries 271
- emergency handling for passwords 265
- field level
 - access types 275
 - DEFINE FIELD command 274
 - entries, addition of 274
 - entries, changing of 274
 - entries, deletion of 274
 - implicit references 274
 - initiation of 274
- file level
 - access 270
 - classifications of 270
 - entries, listing of 272
 - LOGFILE command 272
 - OPENCTL command 270
 - password table 271
 - PRIVDEF parameter 270
 - viewing of 270
- group level
 - entries, addition of 273
 - entries, deletion of 273
 - listing of 273
 - LOGCTL command 273
 - LOGGRP command 273
- group level access to 273
- group level classifications of 273
- in addition to MODEL 204 254
- LOGCTL command 271
- login 251
 - entries, change of 255
 - entries, deletion of 255
 - entries, listing of 254
 - implementation of 252
 - in z/VSE 251
 - LOGCTL command 255
 - LOGFAIL parameter 252
 - LOGTRY parameter 252
 - password table backup 255
 - SYSOPT parameter setting 252
- procedure level
 - access to 276
 - LOGCTL command 276
- record level
 - CREATE command 274
 - INITIALIZE command 274
 - initiation of 274
 - OPENCTL parameter 274
 - override of 274
- removing file-level 272
- terminal level 276
- terminal-level lists 276
- TMASKUPDATE command 116
 - types of 269
 - updating terminal-level 276
 - with password expiration 258
- Security login
 - entries, addition of 255
 - LOGADD parameter 255
 - LOGCTL command 255
 - privilege bytes 253
 - updates, processing of 256
 - updates, storage allocation 256
- security parameters
 - changing the values 261
 - displaying 261
 - setting 260
- SEP parameter 509
- separating transaction and auditing information 297
- separator page 502, 509
- SEQOPT parameter 28
- sequential processing
 - block size 513
 - CMS 512
 - data length 513
 - logical record length 513
 - record format 512
 - SAM 512
 - system requirements (OS) 512
- Server areas 46
- server datasets 169
- server optimization 456
- server sizing 167
- server swapping 166, 169
- Server tables
 - storing request information
- servers
 - allocation 171
 - CKD devices 168, 169
 - FBA devices 167, 168
 - job control 171
- SERVNSA user parameter 109
- SERVNSSZ user parameter 109
- SERVSIZE parameter 24, 29, 46, 167, 169
- SGMTI statistic
 - description and offset 558
- SGMTO statistic
 - description and offset 558
- Since-last statistics 453
 - user 580
- single-user mode (CMS) 105
- SIOSLICE scheduling parameter 141
- SLIC statistic
 - description and offset 558
- SLICEMAX scheduling parameter 141
- SMFLORN parameter 593
- SMFSLRN parameter 593

SMPLS statistic	BLKCFRE 548
description and offset 558	BLKI 549
snap formatter	BLKO 549
improvements 323	BLKRLK 549
SNAPCRAM utility 78, 86	buffer adequacy 436
interacting with the operator 91	BXCHNG 549
JCL 91	BXDELE 549
running 86	BXFINd 549
z/VSE 91	BXFREE 549
SNAPFAIL parameter	BXINSE 549
tracking snaps 324	BXNEXT 549
SNAPFLIM parameter	BXRfND 549
limiting snaps 324	BXSPLI 549
SNAPLIN parameter	CDLWAIT 549
number of CCAMDMP datasets 326	CNCT 549
SORTS statistic	COMMITs 549
description and offset 558	CPU 549
Spare core	DEV10 550
and output definitions 503	DEV11 550
for subsystems 191	DEV12 550
spare core, See SPCORE parameter	DEV13 550
SPCORE	DEV14 550
calculating for ROLL FORWARD facility 367	DEV17 550
SPCORE parameter 24, 29	DEV18 550
VSAM processing 516	DEV19 550
SQLBUFSZ parameter 95, 455	DEV20 550
SQLBUFSZ user parameter 110	DEV23 550
SQLI statistic 455	DEV24 550
description and offset 558	DEV27 550
SQLIQBSZ user parameter 110	DEV28 550
SQLO statistic 455	DEV31 550
description and offset 558	DEV32 550
SQRD statistic	DEV37 550
description and offset 558	DEV38 550
SQWR statistic	DEV49 550
description and offset 558	DEV5 550
SRVSLICE scheduling parameter 141	DEV50 550
Standard deviation	DEV53 550
definition of 308	DEV54 550
START command 116	DEV55 550
START FILE command 195	DEV56 550
START option	DEV57 551
UTILJ utility 395	DEV58 551
START SUBSYSTEM command 195	DEV59 551
START SYSBYSTEm command 195	DEV6 550
start, automatic 203	DEV60 551
STARTIO, See IOS BRANCH ENTRY	DEV61 551
Statistics	DEV62 551
AUDIT 548	DEV63 551
AUDIT204 utility 306	DEV64 551
BACKOUTS 548	DEV65 551
BADD 548	DEV66 551
BCHG 548	DEV67 551
BDEL 548	DEV68 551

DEV69 551
DEV7 550
DEV70 551
DEV71 551
DEV72 551
DEV73 551
DEV74 551
DEV8 550
DEV9 550
DIRRCD 551
DKAR 551
DKPR 551
DKPRF 551
DKRD 551
DKRDL 551
DKRR 551
DKSAWB 551
DKSAWBL 552
DKSAWW 552
DKSAWWL 552
DKSDIR 552
DKSDIRT 552
DKSFBS 552
DKSKIP 552
DKSKIPT 552
DKSRR 552
DKSTBLA 552
DKSTBLB 553
DKSTBLC 553
DKSTBLD 553
DKSTBLF 553
DKSTKQC 553
DKSWRP 553
DKSWRPT 553
DKUPTIME 553
DKWR 553
DKWRL 553
DUMP 553
DUPDTS 553
ECCALL 553
ECCNCT 553
ECCTOUT 553
ECCWAITM 553
ECCWAITS 554
ECDELETE 554
ECLOAD 554
ECMODMAX 554
ECNAMMAX 554
ECTSKMAX 554
ECTWAITM 554
ECTWAITS 554
ERRPDL 554
FBWT 554
FINDS 554

FSCB 554
FSCBSW 554
FTBL 554
GTBL 554
HEAP 554
IN 554
INCMFS 554
INCMIO 554
INCRAM 554
INVMFS 554
INVMIF 555
INVMIO 555
INXX 555
ITBL 555
IXADD 555
IXDEL 555
LKPOST 555
LKWAIT 555
LONGUPDIME 555
LONGUPDTS 555
MOVE 555
MQAPICNT 555
MQAPITIM 555
MQBYTEIN 556
MQBYTEOU 556
MQGETS 556
MQGWTCNT 556
MQGWTSUC 556
MQGWTTIM 556
MQGWTTSP 556
MQHWQU 556
MQHWTASK 556
MQNUMQM 556
MQNUMQU 556
MQPUTS 556
MQWTM 556
NTBL 556
OFFIN 556
OFFOU 556
OUT 556
OUTCMFS 556
OUTCMIO 556
OUTCRAM 556
OUTPB 557
OUTVMFS 557
OUTVMIF 557
OUTVMIO 557
OUTXX 557
PBRSFLT 557
PCPU 557
PDL 557
PNDGTIME 557
PR 557
QTBL 557

- RECADD 557
- RECDEL 557
- RECDS 557
- REDY 557
- reports (AUDIT204) 306
- REQ 557
- REST 557
- RETRYA 557
- RETRYC 557
- RQTM 558
- RSXCOMP 558
- RUNG 558
- SCHDCPU 558
- SCREENS 558
- SGMTI 558
- SGMTO 558
- SLIC 558
- SMPLS 558
- SORTS 558
- SQLI 558
- SQLO 558
- SQRD 558
- SQWR 558
- STBL 558
- STCPU 558
- STDEQ 558
- STIMERS 558
- STPOST 558
- STRECDs 558
- STWAIT 559
- subtype X'08' 573
- SVAC 559
- SVMX 559
- SVPAGES 559
- SVRD 559
- SVWR 559
- SWPG 559
- SWT 559
- system 563, 564, 572
 - performance 563
- system subtype X' 04' 572
- system subtype X'00' 564
- system subtype X'01' 564
- TFMX 559
- TTBL 559
- Type 11 format 588
- type 12 format 590
- Type 13 format 588
- UBUFHWS 559
- UDD 559
- UPDTHME 559
- user 574, 580, 586
 - since-last 580
- user subtype X'04' 586

- USMX 559
- USRS 559
- VTBL 559
- WAIT 560
- WTCFR 560
- WTRLK 560
- WTSV 560
- XTBL 560
- statistics
 - 64-bit layout 9, 46, 149
 - accounting, accuracy of 433
 - APSULDT 548
 - APSY load 180
 - APSYLD 548
 - APSYLDD 548
 - buffer monitoring 435
 - CCASYS file 176
 - DKSRRFND 552
 - dynamic storage allocation 449
 - fast page reads 461
 - file 586, 587
 - partial 587
 - GTBLRS 554
 - GTBLRU 554
 - monitoring of 302
 - storage allocation 449
 - system 563, 571
 - 02 571
 - final 563
 - partial 563
 - System Management Facility (SMF) 592
 - TEMX 559
 - TSMX 559
 - type 10 format 586
 - Type 11 format 588
 - type 13 format 589
 - type 14 format 591
 - type 15 format 591
 - user 575, 576, 581, 584, 585
 - 00 576
 - 01 581, 584, 585
 - 02 576
 - final 575
 - partial statistics 576
 - performance statistics 585
- statistics, Model 204 453
- STATUS command
 - and unsuccessful logins 253
 - report on recovery 369
- status messages
 - for extended quiesce 392
- status reporting
 - suppressing recovery messages 369
- STBL 58, 456

- STBL statistic
 - description and offset 558
- STCPU statistic
 - description and offset 558
- STDEQ statistic
 - description and offset 558
- STEPLIB dataset
 - required for media recovery 386
- STIMERS statistic
 - description and offset 558
- STM 243
- STOP FILE command 195
- STOP option
 - UTILJ utility 395
- STOP SUBSYSTEM command 195
- Storage
 - for password table 256
- storage
 - above the bar 8
 - allocation statistics 449
 - core 449
 - dynamic 449
- storage protection 457
- Store Multiple 243
- STPOST statistic
 - description and offset 558
- stream configurations 411, 412, 415
 - concatenated 416
 - ring 411
 - UTILJ utility 393
- streams
 - switching to next member 288
 - without records and switching 291
- STRECDs statistic 454
 - description and offset 558
- STWAIT statistic
 - description and offset 559
- SUBSYSMGMT 181, 185, 186, 189, 190
 - automatic COMMIT 204
 - automatic login 204
 - automatic logout 204
 - automatic start 203
 - AUTOSYS parameter 196
 - CCASYS file 176, 181
 - CCATEMP file 147
 - CCATEMP requirements 190
 - command line variable 207
 - commands 195
 - communications variable 207
 - debugging 212
 - DICTIONARY interface 181
 - disconnect processing 187
 - error procedure 207
 - error processing 188
 - error variable 208
 - file privileges 213
 - general description 181
 - initialization procedure 207
 - login procedure 207
 - login processing 185
 - long request control 452
 - loop control 204
 - MAXIMUM ITERATIONS option 204
 - multiple procedure files 208
 - NUMLK option 210
 - Operational Parameters screen 202
 - options 191
 - procedure changes 208
 - procedure prefixes 206
 - Procedure Specifications screen 205
 - processing 186
 - main 186
 - record security 212
 - required files 189
 - requirements 189, 190
 - resource locking table 189
 - server tables 190
 - screens, general 196
 - security 210
 - SPCORE requirements 191
 - start privileges 205
 - Subsystem Class User screen 215
 - Subsystem Classes screen 210
 - Subsystem File Use screen 208
 - subsystem startup 185
 - User Definitions screen 213
 - User Matrix screen 223
 - z/OS/XA 185
- SUBSYSMGMT subsystem
 - reviewing 224
- subsystem
 - error messages 197
 - general 197
- Subsystem Access Control Block 76
- subsystem attribute changes
 - limitations 227
- Subsystem Class Users screen
 - client definition 214
- subsystem definitions, See CCASYS
- Subsystem Management facility, See SUBSYSMG-MT
- subsystem procedure
 - handling a blocked refresh 230
- subsystem startup
 - SUBSYSMGMT 185
- Subsystems
 - communications global variable handling 188
 - disabling a subsystem file 184

- error processing 188
- recoverable errors 188
- subsystems 231
 - available at initialization 177
 - enabling a disabled subsystem file 184
 - members (files and permanent groups) 182
 - processing precompiled procedures 230
 - transaction boundaries 353
- subtask affinity
 - definition of 529
 - setting ECISUBS parameter 530
 - setting ECMSUBS parameter 530
- Subtasks
 - statistics 573
- subtasks
 - assignment in ECF 529
- sub-transaction checkpoints 339
 - definition of 337
 - definition restrictions 340
 - eliminating checkpoint time-out 337
 - in a multiuser job 339
 - or transaction checkpoints instead 338
 - using in recovery 338
- SUBTYPE option
 - UTILJ utility 395
- superuser 252
- support for
 - 64-bit architecture 7, 37
 - all stream configurations 290
- SUSPEND SUBSYSTEM command
 - messages 231
- SVAC statistic
 - description and offset 559
- SVCs
 - correct value 592
- SVMX statistic
 - description and offset 559
- SVPAGES statistic 457
 - description and offset 559
- SVRD statistic
 - description and offset 559
- SVWR statistic
 - description and offset 559
- switch processing
 - parallel streams 291
- SWITCH STREAM CCAJRNL command
 - during extended quiesce 289
 - insuring a checkpoint 289
- SWITCH STREAM command
 - concatenated streams 291
 - journals and checkpoints 288
 - parallel streams 290
 - stream types supported 288
 - using 288
- SWPG statistic
 - description and offset 559
- SWT statistic
 - description and offset 559
- SYSLOG ID 132
- SYSLST
 - CCASNAP file 324
- SYSMDUMP datasets
 - allocating 325
- SYSMDUMP facility
 - taking unformatted system dumps 325
- SYSOPT parameter 252
 - CCAAUDIT 305
 - CCAJRNL 299
 - CCASYS 177
 - client subsystem access 177
 - defining runtime actions 25
 - setting for ROLL FORWARD processing 360
 - specifying for IFAM1 in z/VSE 482
 - z/VSE 30
- SYSOPT=X'40' 32
- SYSOPT2 parameter
 - determining XTIO or TIOT 498
- SYSOT parameter
 - UPSI Job Control statement 482
- SYSARM parameter
 - in UPSI Job Control statement 31
- system failures
 - restarting after 367
- system files 181, 283, 306, 320, 325, 346
 - CCAGRP 113, 156
 - CCAJRNL 293
 - CCAPRINT 281, 282
 - CCASERVER 166
 - CCASNAP 320
 - CCASTAT 249, 252
 - CCASYS 176, 181
 - CCATEMP 145
 - CHKPOINT 332, 340, 341
 - CHKPOINT dataset 342
 - CHKPOINT size limitation 342
- System LX
 - usage 77
- System Management Facility (SMF)
 - activation of 593
 - CMS requirements 592
 - logout statistics 592
 - parameters for 593
 - record format 593, 597
 - requirements for use of 591
 - since-last statistics 592
 - z/OS requirements 592
- system manager responsibilities
 - expiring passwords 265

- System programmers
 - consideration for CCATEMP in Storage 151, 167
 - considerations for CCASERVER in Storage 151, 167
- System Resource Manager (SRM) 432
- System statistics
 - partial 563
- system statistics 563
 - final 563

T

- Table pages
 - above the bar 8
- Tables, server, See Server tables
- tape mounts (CMS) 22
- Task Input/Output Table. See TIOT 497
- TBO
 - See also transaction back out 352
- TCB
 - transaction control block 365
- temporary file groups 156
- Temporary work page list (TTBL) 59
- temporary work page list (TTBL) 59
- TEMPPAGE parameter
 - compared to APSYPAGE 178
 - reducing use 179
- TEMX statistic
 - description and offset 559
- TERMBUF user parameter 110
- TERMID parameter 502
- TERMID user parameter 110
- terminal interfaces 65
- terminal-level security 276
 - LOGCTL command 277
 - LOGFILE command 277
 - LOGGRP command 277
 - LOGLST command 277
 - TMASKUPDATE command 277
- Terminals
 - 3270 72
 - teletype 72
- termination, ONLINE 117
- TERMOPT user parameter 110
- TEST command 195
- TFMX statistic
 - description and offset 559
- third-party backup
 - programming 389
- third-party backups
 - managing 388
 - NonStop/204 388
 - sample procedure 391
- third-party support applications
 - processing statistics 9
- TIME parameter 24
- TIMEOUT user parameter 110
- Timer PC
 - use 433
- Timer SVC
 - use 433
- TIMESTOP parameter 29
- tiny load
 - definition of 181
- tiny loads
 - tracking with APSYLD 181
- TIOT
 - dataset limits 497
 - option in ALLOCATE command 498
 - reallocation for RESTART recovery 497
 - with dynamically allocated files 497
- TMASKUPDATE command 116
 - terminal-level security 277
- TODATE option
 - behavior without FROMDATE option 396
 - behavior without times options 396
 - UTILJ utility 394
- TOTIME option
 - behavior without dates options 396
 - behavior without FROMTIME option 396
 - UTILJ utility 394
- TPROCESS, See Program Communication facilities
- TRACEX command 78
- transaction back out
 - See also TBO
 - and AT MOST ONE attribute 356
 - and Subsystem Management facility 353
 - and UNIQUE attribute 356
 - backing out updates 352
 - CCATEMP file 147
 - disabling 356
 - disadvantages of disabling 356
 - during ROLL FORWARD processing 365
 - files and ROLL FORWARD processing 361
 - FOPT setting 356
 - how it works 354
 - log 365
- Transaction Back Out facility
 - CCATEMP 146
- transaction boundaries 352
 - definition of 352
- transaction checkpoints
 - definition of 337
- transaction control blocks (TCB) 365
- transactions

- backing out updates 353
- boundaries 352
- ending 352
- in APSY 353
- Transfer Control, See Program Communication facilities
- TRANSID parameter 502
- translation tables 241
 - modification of 242
- TSMX statistic
 - description and offset 559
- TSO Interface
 - CRAM channel names 93
- TTBL 59
- TTBL statistic
 - description and offset 559

U

- UBUFHWS statistic
 - description and offset 559
- UDD statistic
 - description and offset 559
- unformatted system dumps
 - SYSDUMP facility 325
 - using in z/OS 325
- UNIQUE attribute 356
 - and transaction back out 356
- unposting
 - CPQZ 389
 - QZSIG 390
- unreferenced list
 - definition of 187
- update IDs
 - tracking update units 363
- update unit
 - definition of 360
- update units 334
 - and user hard restarts 364
 - definition of 337, 363
 - determining update ID 368
 - logged in CCAJRNL 361
 - report of ROLL FORWARD processing 369
 - tracking in the audit trail 363
- update units that can be backed out
 - in recovery 363
- update units that cannot be backed out
 - in recovery 363, 365
- updates
 - back out log 354
 - backing out 353
 - identifier 368
 - making permanent 352
 - that cannot back out 353
 - tracking during recovery 379
- Updating data
 - COMMIT statement 352
 - committing transactions 352
 - on remote nodes 352
- UPDTID parameter
 - update identifier 368
- UPDTIME statistic
 - description and offset 559
- UPSI bit settings 30
- UPSI Job Control statement
 - and SYSOPT parameter 482
- USE command 502
 - ROUTER option 502
 - WITH keyword 502
- USE PRINTER command 502, 504
 - z/OS/CMS 502
 - z/VSE 504
- User 0
 - common runtime parameters 26
 - output 281
 - parameter line 15
 - priority scheduling 144
 - z/OS output 282
 - z/VM considerations 283
 - z/VSE output 282
- User 0 input stream
 - z/VM 26
- User 0, See CCAIN file
- User 0, See CCAPRINT file
- User abend code 2749 319
- user hard restart recovery 367
- user hard restarts
 - and update units 364
 - ROLL FORWARD processing 364
- user IDs
 - and LOGLST command 258
 - reinstating 258
 - suspending 258
- User Language
 - CCATEMP file 147
- User Language commands
 - MONITOR ACTIVE 125
- User Language statements
 - COMMIT 352
 - COMMIT RELEASE 352
- user parameters
 - IODEV=39 99
 - LOUTPB
 - use with CICS 543
- user privileges, alteration of 141
- User statistics 574
- Users

- security table 57
- user-written \$functions
 - changes required in 243
- user-written functions 239
- USMX statistic
 - description and offset 559
- USRS statistic
 - description and offset 559
- UTABLE command 46
- UTILC utility
 - description 342
 - JCL 344
 - options 344
 - z/OS JCL 345
- utilities
 - MERGEJ 404
 - UTILC 342
 - UTILJ 393, 400
 - ZBLDTAB 249
 - ZCTLTAB 260
- UTILJ options
 - controlling execution and output 394
- UTILJ REPORT options
 - buffers for 396
- UTILJ utility
 - analyzing problems 400
 - and stream configurations 393
 - applicable return codes 398
 - CMS code example 399
 - date-time combinations behavior 396
 - histogram 396
 - return codes 398
 - usage 393
 - z/OS operating systems JCL 399
 - z/VSE operating system JCL 399

V

- variable table size 48
- variable-format disks 10
- VIEW command 116
 - with dynamic storage 449
- VIEW NUMBUF command 38
- virtual storage
 - above the bar 8
- virtual storage space
 - address assignment 8
- VM
 - and the audit trail 306
 - IUCV interface 97
 - M204 command 101
 - ONLINE processing 18
 - optimization 433

- resource locking 37
- stacking runtime parameters 26
- VM operating system
 - recovery code examples 376
 - statements required for recovery 375
- VMDUMP 324
- VMFSCHNL parameter 98
- VMIFCHNL parameter 98
- VSAM processing 515
 - file loading requirements 517
 - MODEL 204 requirements 515
 - system requirements
 - SPCORE 516
 - STRINGS option 516
- VSE
 - UPSI Job Control statement 31
- VTAM 72, 116
 - APPL for Horizon 97
 - conversion exit routines 606
 - rules 603
 - X3270CHK 604
 - X3270IN 608
 - X3270OUT 606
 - monitoring of 120
 - VTAMOFF command 116
- VTAMOFF command 116
- VTBL 59, 61
- VTBL statistic
 - description and offset 559

W

- WAIT statistic
 - description and offset 560
- Wait types 122, 531
- WAITSCAN scheduling parameter 141
- WARN command 116
- WITH keyword 502
- WTCFR statistic
 - description and offset 560
- WTRLK statistic
 - description and offset 560
- WTSV statistic
 - description and offset 560

X

- X3270CHK routine 604
- X3270OUT routine 606
- XA storage 29
- XDM
 - activating 86
 - definition of 75

- implementing 77
 - monitoring 79
 - performance benefits 76
- XECB version of CRAM 87
- XMEMOPT parameter 29, 75
 - type of CRAM 77
- XMEMSVC parameter 30
- XPCC version of CRAM 87
- XTBL 61
- XTBL statistic
 - description and offset 560
- XTIOT
 - and IOS Branch Entry 498
 - datasets unlimited 497
 - option in ALLOCATE command 498
 - reallocation for RESTART recovery 497
 - with dynamically allocated files 497

Z

- z/Architecture
 - above the bar storage 8
- z/OS
 - address space 432
 - and IFAM1 480
 - AUDIT204 JCL 316
 - DEFINE PRINTER command 502
 - DEFINE PUNCH command 503
 - device types 65
 - directed output 503
 - FREE command 500
 - I/O performance 434
 - IOS BRANCH ENTRY 434
 - keeping the journal 300
 - storage protection 457
 - support for CICS 484
 - System Resource Manager 432
 - USE command 502
 - UTILC utility 345
- z/OS environment
 - User 0 output 282
- z/OS operating system
 - CRAM buffer allocation 85
 - ECF subtasks 520
 - MERGEJ JCL 405
 - recovery JCL 372
 - sample media recovery code 387
 - using unformatted system dumps 325
 - UTILJ JCL 399
- z/OS operating systems
 - perpetual journaling 419
- z/OS/XA 39, 185
 - AMODE 604

- buffers 38
 - control blocks, subsystem 185
 - I/O performance 434
 - IOS BRANCH ENTRY 434
 - VTAM exit routines 604
- z/VM
 - device types 66
- z/VSE
 - AAUDIT file 306
 - and the audit trail 305
 - CCAPPR dataset 507
 - CCAPRINT file 282
 - CCASNAP file 324
 - CCASTAT 250
 - creation of 250
 - CCATEMP file 152
 - console communication 131
 - device types 65
 - directed output 507
 - file group dataset 160
 - optimization 433
 - parallel checkpoint stream 425
 - partition GETVIS 481
 - password table 251
 - reply ID number 132
 - resource locking 36
 - ring/parallel journal stream 423
 - storage considerations 10
 - UPSI bit settings 30
 - USE PRINTER command 504
 - USE PUNCH command 505
 - UTLA utility 306
 - VSAM processing requirements 515
 - z/VSE/POWER 507
- z/VSE CRAM 87, 91
- z/VSE environment
 - User 0 output 282
- z/VSE operating system
 - checkpoint and journal files on disk 373
 - checkpoint and journal files on tape 373
 - CHKPNTD dataset 378
 - considerations for UTILJ reports 398
 - CRAM versions 87
 - deferred update recovery limitation 382
 - IFAM2 92
 - JCL for recovery 372
 - MERGEJ utility code 405, 406
 - recovery-restart from disk 374
 - recovery-restart from tape 375
 - SYSIPT logical unit 25
 - UTILJ JCL 399
- z/VSE/POWER 507
- ZBLDTAB utility 249
- ZCTLTAB parameters

- EXP, PURGE, WARN 265
- viewing 265
- ZCTLTAB utility
 - condition codes 261
 - JCL for z/OS 263
 - modifying CCASTAT 264
 - Password Expiration feature 260
 - running to update CCASTAT 262
 - sample JCL for z/VSE 264
 - setting parameters 265
 - understanding 260