Recovery Utility for VSAM
User Guide

Supporting
Recovery Utility for VSAM version 4.1.00
October 2009
Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

**United States and Canada**

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC SOFTWARE INC</td>
<td>713 918 8800 or</td>
<td>713 918 8000</td>
</tr>
<tr>
<td>2101 CITYWEST BLVD</td>
<td>800 841 2031</td>
<td></td>
</tr>
<tr>
<td>HOUSTON TX 77042-2827</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Outside United States and Canada**

<table>
<thead>
<tr>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>(01) 713 918 8800</td>
<td>(01) 713 918 8000</td>
</tr>
</tbody>
</table>

© Copyright 2009 BMC Software, Inc.

BMC, BMC Software, and the BMC Software logo are the exclusive properties of BMC Software, Inc., are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other BMC trademarks, service marks, and logos may be registered or pending registration in the U.S. or in other countries. All other trademarks or registered trademarks are the property of their respective owners.

CICS, DB2, IMS, RACF, VTAM, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

BMC Software considers information included in this documentation to be proprietary and confidential. Your use of this information is subject to the terms and conditions of the applicable End User License Agreement for the product and the proprietary and restricted rights notices included in this documentation.

**Restricted rights legend**

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC SOFTWARE INC, 2101 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

You can obtain technical support by using the BMC Software Customer Support website or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, see “Before contacting BMC.”

Support website

You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support_home. From this website, you can

■ read overviews about support services and programs that BMC offers
■ find the most current information about BMC products
■ search a database for issues similar to yours and possible solutions
■ order or download product documentation
■ download products and maintenance
■ report an issue or ask a question
■ subscribe to receive proactive e-mail alerts when new product notices are released
■ find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail

In the United States and Canada, if you need technical support and do not have access to the web, call 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:<yourSupportContractID>, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC

Have the following information available so that Customer Support can begin working on your issue immediately:

■ product information
   — product name
   — product version (release number)
   — license number and password (trial or permanent)
■ operating system and environment information
   — machine type
   — operating system type, version, and service pack or other maintenance level such as PUT or PTF
   — system hardware configuration
   — serial numbers
   — related software (database, application, and communication) including type, version, and service pack or maintenance level
■ sequence of events leading to the issue
■ commands and options that you used
■ messages received (and the time and date that you received them)
   — product error messages
   — messages from the operating system, such as file system full
   — messages from related software
License key and password information

If you have questions about your license key or password, contact BMC as follows:

- **(USA or Canada)** Contact the Order Services Password Team at 800 841 2031, or send an e-mail message to ContractsPasswordAdministration@bmc.com.

- **(Europe, the Middle East, and Africa)** Fax your questions to EMEA Contracts Administration at +31 20 354 8702, or send an e-mail message to password@bmc.com.

- **(Asia-Pacific)** Contact your BMC sales representative or your local BMC office.
Contents

About This Book 29

How This Book Is Organized .......................................................... 29
Related Documentation ................................................................. 31
  Online Books ............................................................... 32
  Online Help ................................................................. 32
Conventions .................................................................. 33
  General Conventions ............................................................ 33
  Syntax Statements ............................................................... 35
References to RUV Libraries ...................................................... 36
References to CICS ................................................................. 36
Support for CICS Journals ......................................................... 36

Chapter 1 Introduction to Recovery Utility for VSAM 37

Recovery of Data Sets with RUV .................................................. 39
  Data Set Structures ............................................................... 40
  Recovery Process ............................................................... 41
Recovery Operations .................................................................. 42
  Backout Recovery Operation .................................................... 42
  Forward Recovery Operation .................................................... 42
  Unit-of-Work Recovery Operation .......................................... 43
Types of Backups Needed for a Recovery .................................. 44
  Crisp Backup ................................................................ 44
  Fuzzy Backup ................................................................ 45
  Recognized Backup Formats .................................................. 45
Recovery Process in RUV ............................................................ 46
RUV Features .................................................................. 47
  Unit-Of-Work Recovery .......................................................... 47
  Definition and Creation Of Groups ........................................... 48
  Last Image Accumulation of VSAM File Archives .................. 48
  Support of IAM Disk Management of VSAM Files .............. 48
  RBA0 Detection Support ........................................................ 49
  Automatic Tracking of Recovery Resources .......................... 49
Backup of VSAM Data Sets ......................................................... 49
  Instant Snapshot Copies of VSAM Data Sets ......................... 50
Archiving of CICS Journal Data ................................................. 50
Journaling of VSAM Data Sets Updated in Batch Mode .......... 50
  Automatic Step Backout for Batch Jobs .................................. 51
Forward and Backout Recovery of VSAM Data Sets ............... 51
  Automated Management of RUV Archives and Backups ........ 52
<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
</tr>
<tr>
<td>52</td>
</tr>
<tr>
<td>52</td>
</tr>
<tr>
<td>52</td>
</tr>
<tr>
<td>53</td>
</tr>
<tr>
<td>53</td>
</tr>
<tr>
<td>53</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>57</td>
</tr>
<tr>
<td>57</td>
</tr>
<tr>
<td>59</td>
</tr>
<tr>
<td>59</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>61</td>
</tr>
<tr>
<td>62</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>66</td>
</tr>
<tr>
<td>66</td>
</tr>
<tr>
<td>68</td>
</tr>
<tr>
<td>69</td>
</tr>
<tr>
<td>69</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>72</td>
</tr>
<tr>
<td>73</td>
</tr>
<tr>
<td>74</td>
</tr>
<tr>
<td>75</td>
</tr>
<tr>
<td>75</td>
</tr>
<tr>
<td>75</td>
</tr>
<tr>
<td>76</td>
</tr>
<tr>
<td>76</td>
</tr>
<tr>
<td>77</td>
</tr>
<tr>
<td>79</td>
</tr>
<tr>
<td>79</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>82</td>
</tr>
<tr>
<td>83</td>
</tr>
<tr>
<td>84</td>
</tr>
</tbody>
</table>
Using the TIME_FORMAT Keyword ......................................... 138
Using the UPPER_CASE_TRANSLATION Keyword ..................... 138
Using the USE_AUTOJOURNAL Keyword ................................ 139
Using the WORK_DISK Keyword ........................................... 141
Using the WORK_TAPE Keyword ............................................ 141
Using the XBMID Keyword .................................................. 141
Viewing Default Settings ...................................................... 142
Using RUV Default Values .................................................... 142
Coding SET DEFAULT Statements ........................................... 143
Coding SET Statements ........................................................ 144
Coding RESET DEFAULT Statements ........................................ 145
Working with VSAM File Repository Records .............................. 146
Using the ADD VSAM_FILE Command ..................................... 146
Coding ADD VSAM_FILE Statements ....................................... 146
Using the VSAM_FILE Keyword (ADD) .................................. 147
Using the SELECTION_EXIT Keyword (ADD) ............................ 147
Using the APPLID_FILEID Keyword (ADD) .............................. 147
Using Other VSAM_FILE Keywords (ADD) .............................. 148
Using the UPDATE VSAM_FILE Command ................................. 148
Coding UPDATE VSAM_FILE Statements ................................ 148
Using the VSAM_FILE Keyword (UPDATE) .............................. 149
Using the SELECTION_EXIT Keyword (UPDATE) ....................... 149
Using the APPLID_FILEID Keyword (UPDATE) ......................... 149
Using Other VSAM_FILE Keywords (UPDATE) ......................... 149
Using the DELETE VSAM_FILE Command ................................. 150
Coding DELETE VSAM_FILE Statements ................................ 150
Using the VSAM_FILE Keyword (DELETE) ............................... 150
Chapter 4 Using the Batch Journaling Facility ........................... 151
Overview of Rule Sets ........................................................... 154
Levels of Rule Sets ................................................................ 154
Rule Set Elements ................................................................. 154
Requirements for Rule Set Element Names ............................... 156
Order of Rule Set Entries ....................................................... 157
Commands for Use with Rule Set Entries ................................. 158
Enabling Recovery of VSAM Files ............................................ 159
Using the ADD VSAM_SET Command ..................................... 159
Using the VSAM_RULE Keyword ............................................ 159
Coding ADD VSAM_SET Statements ....................................... 160
Identifying Jobs, Programs, and Recovery Actions ...................... 161
Using the ADD JOB_SET Command ........................................ 161
Using the JOB_RULE Keyword .............................................. 161
Using the LOG Keyword ...................................................... 162
Using the JOURNAL Keyword ................................................ 162
Using the JOURNAL_ABEND Keyword ................................... 163
Using the JOB_ABEND Keyword .......................................... 163
Using the JOB_JCL Keyword ............................................... 163
Using the INTERNAL_READER Keyword ............................... 164
Coding ADD JOB_SET Statements ......................................... 164
Using the ARCHIVE_OUT Keyword ..................................................... 232
Using the UNIT_OF_WORK_OUT Keyword ......................................... 232
Using Optional Output Keywords ...................................................... 233
Coding ARCHIVE LOG_STREAM_IN Statements .................................. 233

Printing Log Stream Data ............................................................... 234
Using the PRINT_LOG_STREAM_IN Command .................................... 234
Using the LOG_STREAM_IN Keyword ............................................... 234
Using the DELETE Keyword ........................................................... 234
Using the START_TIME Keyword .................................................... 234
Coding PRINT LOG_STREAM_IN Statements ...................................... 235

Managing Log Stream Records ....................................................... 236
Using the ADD LOG_STREAM Command ............................................ 236
Using the UPDATE LOG_STREAM Command ....................................... 236
Using the DELETE LOG_STREAM Command ....................................... 237
Using the REPORT LOG_STREAM Command ...................................... 237
Using Optional Keywords ............................................................ 238
Coding ADD LOG_STREAM Statements .............................................. 238
Coding UPDATE LOG_STREAM Statements ....................................... 238
Coding DELETE LOG_STREAM Statements ....................................... 238

Managing the Transaction Server Log of Logs .................................. 239
Using the REGISTER LOG_OF_LOGS Command .................................. 239
Using the ADD LOG_OF_LOGS Command .......................................... 240
Using the UPDATE LOG_OF_LOGS Command ..................................... 240
Using the DELETE LOG_OF_LOGS Command ..................................... 240
Using the REPORT LOG_OF_LOGS Command .................................... 241
Using the LOG_OF_LOGS Keyword .................................................. 241
Using the DELETE Keyword .......................................................... 241
Using the DAYS Keyword ............................................................. 242
Using the START_TIME Keyword .................................................... 242
Using Optional Keywords ............................................................ 242
Coding ADD LOG_OF_LOGS Statements .......................................... 242
Coding DELETE LOG_OF_LOGS Statements ..................................... 243
Coding REGISTER LOG_OF_LOGS Statements .................................. 243
Coding UPDATE LOG_OF_LOGS Statements ..................................... 243

Chapter 6  Working with Archives .................................................. 245

Overview of Archive Management .................................................. 247
Accumulating Archive Data Sets ...................................................... 248
Separating Archives ................................................................. 248
Merging Archives and Combining Data ............................................. 248
Retaining and Purging Input Archives .............................................. 249
Setting Up the Accumulation Job .................................................... 249
Using the ARCHIVE ACCUM Command ......................................... 250
Using the INVENTORY_ONLY Keyword .......................................... 251
Using the ARCHIVE_IN Keyword .................................................. 251
Using the SCRATCH Keyword ........................................................ 252
Using the ACCUM Keyword ........................................................ 254
Using the CONSOLIDATED Keyword .............................................. 254
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the VSAM_FILE Keyword</td>
<td>254</td>
</tr>
<tr>
<td>Using the VSAM_GROUP Keyword</td>
<td>254</td>
</tr>
<tr>
<td>Using the UNSELECTED_VSAM_FILES Keyword</td>
<td>254</td>
</tr>
<tr>
<td>Using the RETAIN_IMAGE Keyword</td>
<td>255</td>
</tr>
<tr>
<td>Using the LAST_IMAGE Keyword</td>
<td>255</td>
</tr>
<tr>
<td>Using the ARCHIVE_OUT Keyword</td>
<td>256</td>
</tr>
<tr>
<td>Suggested Usage and Example 1</td>
<td>256</td>
</tr>
<tr>
<td>Suggested Usage and Example 2</td>
<td>259</td>
</tr>
<tr>
<td>Copying Archive Data Sets</td>
<td>260</td>
</tr>
<tr>
<td>Using the ARCHIVE ARCHIVE_IN Command</td>
<td>260</td>
</tr>
<tr>
<td>Using the ARCHIVE_IN Keyword</td>
<td>260</td>
</tr>
<tr>
<td>Using the ARCHIVE_OUT Keyword</td>
<td>261</td>
</tr>
<tr>
<td>Coding ARCHIVE ARCHIVE_IN Statements</td>
<td>261</td>
</tr>
<tr>
<td>Registering Archive Data Sets</td>
<td>262</td>
</tr>
<tr>
<td>Using the REGISTER ARCHIVE_IN Command</td>
<td>262</td>
</tr>
<tr>
<td>Using the ARCHIVE_IN Keyword</td>
<td>262</td>
</tr>
<tr>
<td>Using Optional Keywords</td>
<td>262</td>
</tr>
<tr>
<td>Coding REGISTER ARCHIVE_IN Statements</td>
<td>263</td>
</tr>
<tr>
<td>Updating Archive Records</td>
<td>264</td>
</tr>
<tr>
<td>Using the UPDATE ARCHIVE_FILE Command</td>
<td>264</td>
</tr>
<tr>
<td>Using the ARCHIVE_FILE Keyword</td>
<td>265</td>
</tr>
<tr>
<td>Using the FILTER_BY Keyword</td>
<td>265</td>
</tr>
<tr>
<td>Using the REGISTRATION_TIME Keyword</td>
<td>265</td>
</tr>
<tr>
<td>Using Optional Keywords</td>
<td>266</td>
</tr>
<tr>
<td>Coding UPDATE ARCHIVE_FILE Statements</td>
<td>266</td>
</tr>
<tr>
<td>Deleting Archive Records</td>
<td>268</td>
</tr>
<tr>
<td>Using the DELETE ARCHIVE_FILE Command</td>
<td>268</td>
</tr>
<tr>
<td>Using the ARCHIVE_FILE Keyword</td>
<td>268</td>
</tr>
<tr>
<td>Using the FILTER_BY Keyword</td>
<td>268</td>
</tr>
<tr>
<td>Using the SCRATCH Keyword</td>
<td>269</td>
</tr>
<tr>
<td>Coding DELETE ARCHIVE_FILE Statements</td>
<td>269</td>
</tr>
<tr>
<td>Reporting Archive Information in the Repository</td>
<td>270</td>
</tr>
<tr>
<td>Using the REPORT ARCHIVE_FILE Command</td>
<td>270</td>
</tr>
<tr>
<td>Using the ARCHIVE_FILE Keyword</td>
<td>270</td>
</tr>
<tr>
<td>Using the FILTER_BY Keyword</td>
<td>271</td>
</tr>
<tr>
<td>Coding REPORT ARCHIVE_FILE Statements</td>
<td>271</td>
</tr>
<tr>
<td>Archiving RPCV Journals</td>
<td>272</td>
</tr>
<tr>
<td>Using the ARCHIVE RPCV_IN Command</td>
<td>272</td>
</tr>
<tr>
<td>Using the RPCV_IN Keyword</td>
<td>272</td>
</tr>
<tr>
<td>Using the COMPLETE_COPY Keyword</td>
<td>272</td>
</tr>
<tr>
<td>Using Other Keywords</td>
<td>273</td>
</tr>
<tr>
<td>Coding ARCHIVE RPCV_IN Statements</td>
<td>273</td>
</tr>
<tr>
<td>Printing Archive Data</td>
<td>274</td>
</tr>
<tr>
<td>Using the PRINT ARCHIVE_IN Command</td>
<td>274</td>
</tr>
<tr>
<td>Using the ARCHIVE_IN Keyword</td>
<td>274</td>
</tr>
<tr>
<td>Using the COUNT Keyword</td>
<td>274</td>
</tr>
<tr>
<td>Using the START_TIME keyword</td>
<td>275</td>
</tr>
<tr>
<td>Using the STOP_TIME keyword</td>
<td>275</td>
</tr>
<tr>
<td>Using the SELECTION_EXIT keyword</td>
<td>275</td>
</tr>
<tr>
<td>Coding PRINT ARCHIVE_IN Statements</td>
<td>276</td>
</tr>
</tbody>
</table>
Chapter 7  Working with Backups 277

Backup Overview .......................................................... 279
Backup Methods ......................................................... 279
Backup Strategies ....................................................... 282
Backup Registration Records in the RUV Repository .............. 283
Considerations for Backups ............................................ 284
Integrating RUV in an Existing Backup Strategy .................. 285
Controlling Backup Environment Options .......................... 286
Creating RUV Backups .................................................. 288
  Overview of the RUV Backup Function .............................. 288
  Using the RUV Backup Function .............................. 289
  Coding JCL for RUV Backups of Individual VSAM Files .... 290
  Coding JCL for RUV Backups of Individual Sequential Files 294
  Coding JCL for RUV Backups of VSAM Groups ................. 294
  Coding JCL for RUV Backups of Sequential Groups .......... 297
  RUV Backup Examples ............................................... 298
Creating BWO Backups ................................................. 300
  Overview of BWO Backups ........................................... 300
  Using the BWO Feature ............................................... 301
Creating Instant Snapshot Copies .................................. 302
  Overview of Instant Snapshot Copies ......................... 302
  Creating an Instant Snapshot Copy .............................. 303
  Working with VSAM Name Models ................................. 304
  Coding JCL to Create Instant Snapshot Copies ............... 307
  Instant Snapshot Copy Examples .................................. 311
Using the RBA0 Backup Detection Feature ........................ 312
  Overview of the RBA0 Backup Detection Feature ............. 312
  Identifying an RBA0 Backup ........................................ 312
  Implementing the RBA0 Backup Detection Feature ............ 313
Using SMS Backups and IDCAMS REPRO Copies ................. 314
  Automatically Using SMS Backups .................................. 314
  Automatically Using IDCAMS REPRO Copies ................... 314
Creating External Backups ......................................... 315
  Overview of External Backup Support .......................... 315
  Implementing External Backup Support ......................... 317
  Example of External Backup Support ............................ 318
Working with Job JCL Records for Backup and Restore Processes 327
  Overview of Job JCL Records ....................................... 327
  Creating a Job JCL Record with the ISPF Interface ........... 327
  Creating a Job JCL Record with the RUVZSM0 Utility ....... 329
  Coding JCL to Work with Job JCL Records .................... 329
Storing Information into a File ........................................ 332
  Coding the STORE INTO_FILE Command ....................... 332
  Example of STORE INTO_FILE Command Usage  .............. 333
Submitting Job JCL Records for Execution ......................... 334
  Overview of the Submit Function ................................. 334
  Submitting a Job JCL Record with the ISPF Interface ........ 334
  Submitting a Job JCL Record with the RUVZSM0 Utility .... 341
  Coding JCL to Submit Job JCL Records for Execution ....... 341
Chapter 8  Creating Reports  373

Reporting RUV Data with ISPF Panels 375
  VSAM DSN List  376
  Applid/Fileid List  378
  VSAM DSN Detail - Base Cluster Panel  379
  DSN Detail - Applid/Fileid List Panel  380
  DSN Detail - User Recovery Exit List Panel  381
  DSN Detail - Comments Pop-up  382
  DSN Detail - Path List Panel  383
  DSN Detail - Backup List Panel  384
  Delete/Define Statements in ISPF Edit Panel  385
  Archive DSN List  387
  Backup File List  389
  Log of Logs List  391
  Groups / Output / VSAM Models  393
  Subsystem / Ruleset Information  395
  Message Lookup  398
  RUV Job Monitoring  400

Reporting RUV Data in Batch  401
  Selecting a Batch Report Level  402
  Obtaining RUV Batch Reports  403
  ARCHIVE_FILE Report  405
  APLID Report  407
  BACKUP_FILE Report  408
  BACKUP_METHOD Report  409
  DEFAULT Report  410
  EXTERNAL_VENDOR Report  411
  INTERNAL_READER Report  412
## Chapter 9 Reorganizing VSAM Files

**Overview**  
438

Using the RUV REORGANIZE Function  
439

Reorganizing Individual VSAM Files  
440

Reorganizing VSAM Groups  
443

## Chapter 10 Restoring Data Sets

**Overview**  
448

RUV Actions for Restore  
448

Automatic Restore during Forward Recovery  
448

NEW NAME Restore  
449

RUV CURRENT BACKUP Restore  
449

Point-in-Time Restore  
449

Non-RUV Backups  
450

Alternate Indexes  
450

Number of Reader Tasks  
450

SMS Backups  
451

IDCAMS Delete/Define Operations for VSAM Files  
451

Restoring a Backup-While-Open Backup  
451

Forward Recovery and the RBA0 Backup Detection Feature  
452

Summary of Forward Recovery Actions  
452

Current Backup Specified  
454

Backup Data Set Name Specified  
454

Recovery Time Specified with No Archives or Backups Specified  
454

Recovery Time and Archives Specified with No Backups Specified  
455

Archives Specified with No Recovery Time or Backups Specified  
456
Restoring Files from Backups ..................................................... 457
Running the Restore Job .......................................................... 457
Using the RESTORE VSAM_FILE Command ................................ 457
Using the RESTORE SEQ_FILE Command .................................. 458
Using the VSAM_FILE Keyword ................................................. 458
Using the SEQ_FILE Keyword ................................................... 459
Using the NEW_NAME KEYWORD ............................................. 459
Using the START_TIME keyword .............................................. 460
Using the STOP_TIME keyword ............................................... 460
Using the AIX Keyword for VSAM Files .................................... 460
Using the BACKUP_IN Keyword .............................................. 461
Coding RESTORE VSAM_FILE Statements ................................. 462
Coding RESTORE SEQ_FILE Statements ................................... 462

Chapter 11  Recovering VSAM Data Sets ................................. 463
Overview of Recovery on a Local System ................................. 465
Forward Recovery ................................................................. 465
Unit-of-Work Recovery ......................................................... 468
Backout Recovery ................................................................. 471
Sort Parameters ................................................................. 472
Using RUV for Forward Recovery ........................................... 473
Running an RUV Forward Recovery ......................................... 473
Using the RECOVER FORWARD Command ............................... 473
Coding RECOVER FORWARD Statements ................................. 474
Using the AIX Keyword ........................................................ 476
Using the ARCHIVE_IN Keyword ......................................... 477
Using the BACKUP_IN Keyword ............................................ 478
Using the BUFND Keyword ................................................... 478
Using the BUFINI Keyword ................................................... 478
Using the INVENTORY_ONLY Keyword ................................... 479
Using the LSR_POOL Keyword .............................................. 479
Using the NEW_NAME Keyword ............................................ 480
Using the RECOVERY_MODE Keyword ................................... 480
Using the SELECTION_EXIT Keyword .................................... 480
Using the START_TIME Keyword ......................................... 481
Using the STOP_TIME Keyword ........................................... 481
Using the SUPPRESS_RECOVERY_ERROR Keyword ................. 482
Using the VSAM_FILE Keyword ............................................ 482
Using RUV for UOW Recovery .................................................. 483
Running an RUV UOW Recovery ............................................ 483
Using the RECOVER COMPLETED_UNIT_OF_WORK Command .... 483
Coding the RECOVER COMPLETED_UNIT_OF_WORK Statements .... 484
Using the ARCHIVE_IN Keyword ......................................... 485
Using the BACKUP_IN Keyword ............................................ 485
Using the INVENTORY_ONLY Keyword ................................... 486
Using the REPORT UNIT_OF_WORK Command .......................... 486
Using the START_TIME Keyword ......................................... 486
Using the STOP_TIME Keyword ........................................... 487
Using the UNIT_OF_WORK_IN Keyword ................................ 487
Phase 1—Preparing for Disaster Recovery .......................................................... 526
  Establishing RUV Groups ................................................................. 526
  Obtaining Backups of Assets .............................................................. 527
  Verifying the Recovery Process ............................................................ 528
Phase 2—Performing Disaster Recovery ....................................................... 529
  Restoring the RUV Repository and Starting RUV .................................... 529
  Understanding the Potential for Lost Data After Recovery ..................... 530
Phase 3—Evaluating the Disaster Recovery .................................................. 531

Chapter 13 Working with BMC Software Subsystems .................................. 533

BMC Software Subsystems ................................................................. 535
  BMCP ................................................................. 535
  BCSS ................................................................. 536
  Multiple BCSSs ............................................................. 536
  Subsystem Procedure Names ............................................................ 537
  Subsystem Identifiers ................................................................. 537
  Specifying General BMC Subsystem Parameters .................................. 537
  CPU Authorization Passwords .......................................................... 538
BMC Software Subsystem (BMCP) ......................................................... 539
  BMCP Command Summary ............................................................ 539
  BMC Primary Subsystem Parameters ................................................ 540
  Starting the BMCP ........................................................................ 541
  Displaying BMCP Status .................................................................. 541
  Stopping the BMCP ........................................................................ 542
BMC Consolidated Subsystem (BCSS) ....................................................... 543
  BCSS Subsystem ID ........................................................................ 543
  BCSS Command Format .................................................................... 543
  Controlling BCSS Using the z/OS Console .......................................... 544
  Using the BCSS Commands Data Set ................................................ 544
  BCSS Command Summary ................................................................ 544
  Starting the BCSS ........................................................................... 545
  Specifying BMC Consolidated Subsystem Parameters ......................... 545
  Initializing the BCSS RUV Components ............................................. 548
  Displaying the Status of the BCSS Address Space .............................. 549
  Activating the RUV Rule Set .............................................................. 549
  Stopping the RUV BCSS Address Space ............................................ 550
  Requesting RUV Help ..................................................................... 550
  Enabling RUV Activity .................................................................... 550
  Stopping RUV Activity Normally ...................................................... 551
  Stopping RUV Activity Immediately ................................................ 552
  Displaying the Status of RUV Components ....................................... 552
  Changing Ownership of the BWO Coordinator .................................... 553
  Displaying RUV Diagnostic Information ............................................. 553
  Using the VALIDATE SUBSYSTEM Command ................................ 554
  Controlling the Subsystem with Subsystem Detail Panel ....................... 555
Chapter 14  Maintaining the RUV Repository 557

REGISET Structure and Use .................................................. 559
  Registration Data Sets ...................................................... 559
  Primary Registration Data Sets ........................................... 559
  Duplex Registration Data Sets ........................................... 560
  REGISET Records ............................................................ 561
  REGISET Serialization Process .......................................... 561
  Size of the Repository ..................................................... 562
Registration Data Set Recovery ........................................... 564
  Recovering a Primary Registration Data Set ......................... 564
  Recovery with Another Primary Registration Data Set ............. 564
  Recovery with a Duplex Registration Data Set ...................... 565
  Recovering from a Backup Copy ......................................... 566
  Recovering a Duplex Registration Data Set .......................... 567
  Recovering an RUV Repository Using SMF Data ...................... 568
REGISET Analysis and Backups ............................................ 572
Backing Up and Restoring RUV Repository Data .......................... 573
  Using the BACKUP REPOSITORY Command ............................. 573
  Using the BACKUP_OUT Keyword ........................................ 574
  Using the RESTORE REPOSITORY Command ......................... 574
  Using the BACKUP_IN Keyword ......................................... 574
  Using the START_TIME keyword ........................................ 575
  Using the STOP_TIME keyword ......................................... 575
  Using the MERGE Keyword .............................................. 575
  Coding BACKUP REPOSITORY Statements .............................. 576
  Coding RESTORE REPOSITORY Commands ............................. 576
Printing Repository Data .................................................... 577
  Using the PRINT REPOSITORY Command ................................ 577
  Using the COUNT Keyword ................................................ 577
  Coding PRINT REPOSITORY Statements ................................ 578
Purging Repository Records ............................................... 579
  Selecting the Purge Technique and Value ............................. 579
  Performing the Purge Procedure ....................................... 580
  Setting the Purge Value at System, Job, and File Levels .......... 580
  Using Commands to Set Purge Values ................................. 581
  Understanding the One-Day Rule ....................................... 581
  Purge Processing Actions .............................................. 581
  Obsolete APPLID/FILEID References .................................. 582
  Backups ........................................................................ 583
  Orphaned Backups ......................................................... 584
  Archives ....................................................................... 584
  Purge Reports. ................................................................ 585
  Purge Output File ............................................................ 588
  Backup before Purge Processing ......................................... 588
  Using the PURGE REPOSITORY Command ............................... 588
Validating the Repository .................................................... 591
  Coding the VALIDATE REPOSITORY Command .......................... 591
Improving Repository Performance ......................................... 591
  Exporting and Importing the Repository ................................ 592
  Reorganizing the Repository ............................................. 594
## Chapter 15  Using VSAM Groups

- **Overview of VSAM Groups** ......................................................... 596
- **Group Definition** .................................................................. 597
- **Group Use** ............................................................................ 597
- **Group Backups** ..................................................................... 597
- **Group Restore** ....................................................................... 598
- **Group Recover** ....................................................................... 598
- **Groups in Rule Sets** .............................................................. 598
- **Group Maintenance** ............................................................... 599
- **Group Archives** ..................................................................... 599
- **Group Reports** ....................................................................... 600
- **Group Purge** .......................................................................... 600
- **Group Update** ........................................................................ 600

### Defining Groups

- **Using the ADD VSAM_GROUP_DEFINITION Command** .............. 601

### Using Groups

- **Using the BACKUP VSAM_GROUP Command** .......................... 603
- **Using the RESTORE VSAM_GROUP Command** ......................... 605
- **Using the RECOVER FORWARD VSAM_GROUP Command** ........ 607
- **Using the RECOVER UOW VSAM_GROUP Command** ............... 609
- **Using the RECOVER BACKOUT VSAM_GROUP Command** ....... 610
- **Using the ARCHIVE ACCUM VSAM_GROUP Command** ............ 611
- **Using the ADD VSAM_SET Command** .................................... 613

### Maintaining Groups

- **Using the ADD OUTPUT_MODEL Command** ............................ 615
- **Using the UPDATE OUTPUT_MODEL Command** ......................... 617
- **Using the DELETE OUTPUT_MODEL Command** ........................ 617
- **Using the REPORT OUTPUT_MODEL Command** ......................... 617
- **Using the UPDATE VSAM_GROUP_DEFINITION Command** ....... 618
- **Using the DELETE VSAM_GROUP_DEFINITION Command** ......... 618
- **Using the DELETE VSAM_GROUP Command** ............................. 619
- **Using the UPDATE VSAM_GROUP Command** ........................... 619
- **Using the REPORT VSAM_GROUP_DEFINITION Command** ....... 620
- **Using the REPORT VSAM_GROUP Command** ............................ 621
- **Using the PURGE REPOSITORY VSAM_GROUP Command** ........ 621

## Chapter 16  Using Sequential Groups

- **Overview of Sequential Groups** ............................................. 624
- **Group Definition** .................................................................. 624
- **Group Use** ............................................................................ 624
- **Group Backups** ..................................................................... 624
- **Group Restore** ....................................................................... 625
- **Group Maintenance** ............................................................... 625
- **Group Reports** ....................................................................... 625
- **Group Purge** .......................................................................... 626
- **Group Update** ........................................................................ 626
Defining Groups ................................................................. 627
  Using the ADD SEQ_GROUP_DEFINITION Command .......... 627
  Coding the ADD SEQ_GROUP_DEFINITION Command ....... 627
Using Groups ................................................................. 629
  Using the BACKUP SEQ_GROUP Command .................... 629
  Using the RESTORE SEQ_GROUP Command ................. 631
Maintaining Groups ......................................................... 633
  Using the UPDATE SEQ_GROUP_DEFINITION Command .... 633
  Using the DELETE SEQ_GROUP_DEFINITION Command .... 633
  Using the DELETE SEQ_GROUP Command ............... 634
  Using the UPDATE SEQ_GROUP Command .................. 634
  Using the REPORT SEQ_GROUP_DEFINITION Command ..... 635
  Using the REPORT SEQ_GROUP Command .................. 636
  Using the PURGE REPOSITORY SEQ_GROUP Command .... 636

Appendix A  Solving Problems 637

Collecting Problem Information .................................... 638
Diagnostic Steps ......................................................... 639
Displaying PTF Details .................................................. 641
Displaying Environment Information with the RUVZSM0 Utility 642
Displaying Environment Information with the RUV ISPF Interface 643

Appendix B  Migrating from RPCV to RUV 645

Overview of Migration from RPCV to RUV ......................... 646
Using the RUVZCJC Utility ........................................... 647
Using the RUVZCDA Utility ........................................... 649
  Creating Data Set, Group Definition, and User Variable Statements .... 650
  Defining Data Sets, Group Definitions, and User Variables to RUV ... 652
Identifying Jobs That Use RPCV ...................................... 653
RPCV-to-RUV Migration Considerations ......................... 654
  Supported File Structures .................................... 654
  Repository Information for VSAM File Recovery ........... 654
  System Requirements ........................................... 654
  Rule Sets ......................................................... 655
  Security ....................................................... 655
  ISPF Interface ................................................. 656
Using BMC Software Subsystems .................................. 656
  Batch Journaling and Job Step Logging .................... 656
Back Up Master Data Sets .............................................. 657
Purge Processing .......................................................... 658
Archiving with RUV ...................................................... 658
RUV Recovery ............................................................. 658
Creating RUV Reports ................................................... 659
Miscellaneous Items ..................................................... 659
# Figures

<table>
<thead>
<tr>
<th>Figure Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recovery Model for Planned Business Continuity</td>
<td>40</td>
</tr>
<tr>
<td>RUV System Flow</td>
<td>41</td>
</tr>
<tr>
<td>Backout Recovery Process</td>
<td>42</td>
</tr>
<tr>
<td>Forward Recovery Process</td>
<td>42</td>
</tr>
<tr>
<td>Unit Of Work Recovery Process</td>
<td>43</td>
</tr>
<tr>
<td>Crisp Backup</td>
<td>44</td>
</tr>
<tr>
<td>Fuzzy Backup</td>
<td>45</td>
</tr>
<tr>
<td>Recovery from a Backup</td>
<td>46</td>
</tr>
<tr>
<td>Batch Journaling Facility Process</td>
<td>56</td>
</tr>
<tr>
<td>Rule Set Relationships</td>
<td>62</td>
</tr>
<tr>
<td>Archive to Archive Accumulation Process</td>
<td>66</td>
</tr>
<tr>
<td>Sample Accumulation Strategy</td>
<td>67</td>
</tr>
<tr>
<td>RUVZSM0 Execution JCL</td>
<td>79</td>
</tr>
<tr>
<td>Help for Command Errors</td>
<td>84</td>
</tr>
<tr>
<td>Help for Keyword Errors</td>
<td>84</td>
</tr>
<tr>
<td>Help for Keyword Value Errors</td>
<td>85</td>
</tr>
<tr>
<td>Help for Variable Substitution Errors</td>
<td>86</td>
</tr>
<tr>
<td>Subsystem List Pop-up Window</td>
<td>114</td>
</tr>
<tr>
<td>RUV Options Panel</td>
<td>115</td>
</tr>
<tr>
<td>Display Filters Popup Window</td>
<td>119</td>
</tr>
<tr>
<td>Date Conversion Popup Window</td>
<td>120</td>
</tr>
<tr>
<td>TOD Clock Date/Time Conversion Popup Window</td>
<td>121</td>
</tr>
<tr>
<td>User Preferences Panel</td>
<td>122</td>
</tr>
<tr>
<td>SET DEFAULT Command Example</td>
<td>124</td>
</tr>
<tr>
<td>SET Command Example</td>
<td>125</td>
</tr>
<tr>
<td>Sample ARCHIVE SUMMARY Report with USE_AUTOJOURNAL</td>
<td>140</td>
</tr>
<tr>
<td>Rule Set Construction</td>
<td>155</td>
</tr>
<tr>
<td>REPORT SYSTEM_VARIABLES Command Output</td>
<td>178</td>
</tr>
<tr>
<td>Sample RUVSTAT Printout</td>
<td>190</td>
</tr>
<tr>
<td>Sample Rule Set $DEFAULT</td>
<td>194</td>
</tr>
<tr>
<td>Modified Rule Set $DEFAULT</td>
<td>196</td>
</tr>
<tr>
<td>Rule Set Build and Activation Flow</td>
<td>197</td>
</tr>
<tr>
<td>TRY Command Summary-Level Results</td>
<td>202</td>
</tr>
<tr>
<td>TRY Command Detail-Level Results</td>
<td>202</td>
</tr>
<tr>
<td>TRY Command Full-Level Results</td>
<td>203</td>
</tr>
<tr>
<td>Subsystem Detail Panel</td>
<td>205</td>
</tr>
<tr>
<td>Ruleset List Panel</td>
<td>206</td>
</tr>
<tr>
<td>Show Popup Window</td>
<td>207</td>
</tr>
<tr>
<td>File Menu Options on Action Bar</td>
<td>208</td>
</tr>
<tr>
<td>Ruleset in ISPF Edit Panel</td>
<td>209</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Activate the Selected Rule</td>
<td>210</td>
</tr>
<tr>
<td>Key Assignments</td>
<td>210</td>
</tr>
<tr>
<td>RUV Ruleset/Group/Model Templates Panel</td>
<td>211</td>
</tr>
<tr>
<td>CICS File Definitions for CICS TS Environments</td>
<td>218</td>
</tr>
<tr>
<td>Archive Accumulation Process</td>
<td>250</td>
</tr>
<tr>
<td>Usage Example of ACCUM Process</td>
<td>257</td>
</tr>
<tr>
<td>Accumulation Example 1</td>
<td>257</td>
</tr>
<tr>
<td>Accumulation Example 2</td>
<td>259</td>
</tr>
<tr>
<td>UPDATE ARCHIVE_FILE statement with REGISTRATION_TIME keyword</td>
<td>267</td>
</tr>
<tr>
<td>BACKUP VSAM_IN Model JCL</td>
<td>290</td>
</tr>
<tr>
<td>BACKUP VSAM_GROUP Model JCL</td>
<td>295</td>
</tr>
<tr>
<td>BACKUP Examples</td>
<td>298</td>
</tr>
<tr>
<td>Instant Snapshot Copy JCL</td>
<td>308</td>
</tr>
<tr>
<td>Example ADD JOB_JCL Command for Backups</td>
<td>319</td>
</tr>
<tr>
<td>Example ADD JOB_JCL Command for Restores</td>
<td>320</td>
</tr>
<tr>
<td>Example Generated Backup JCL</td>
<td>322</td>
</tr>
<tr>
<td>Example Generated Restore JCL</td>
<td>324</td>
</tr>
<tr>
<td>ISPF Edit Panel for Job JCL</td>
<td>328</td>
</tr>
<tr>
<td>ADD JOB_JCL Model JCL for a Backup Job</td>
<td>329</td>
</tr>
<tr>
<td>VSAM Sphere External Backup / Restore Panel</td>
<td>335</td>
</tr>
<tr>
<td>VSAM Sphere External Backup / Restore - Rule_Set Names Panel</td>
<td>336</td>
</tr>
<tr>
<td>VSAM Sphere External Backup / Restore - VSAM File Names Panel</td>
<td>337</td>
</tr>
<tr>
<td>VSAM Sphere External Backup / Restore - Job_JCL Names Panel</td>
<td>338</td>
</tr>
<tr>
<td>ISPF Edit Panel Displaying Job JCL Record Contents</td>
<td>339</td>
</tr>
<tr>
<td>ISPF Edit Panel Displaying Generated JCL</td>
<td>340</td>
</tr>
<tr>
<td>SUBMIT BACKUP Model JCL for a Backup Job</td>
<td>341</td>
</tr>
<tr>
<td>REPORT and DELETE EXTERNAL_JOB Command Syntax</td>
<td>343</td>
</tr>
<tr>
<td>EXTERNAL_JOB Report</td>
<td>344</td>
</tr>
<tr>
<td>COPY BACKUP Command Example JCL</td>
<td>346</td>
</tr>
<tr>
<td>COPY BACKUP Command Syntax for a VSAM Group</td>
<td>347</td>
</tr>
<tr>
<td>COPY BACKUP Command Syntax for a VSAM File</td>
<td>347</td>
</tr>
<tr>
<td>VSAM DSN List Panel</td>
<td>376</td>
</tr>
<tr>
<td>Filter Pop-Up Panel</td>
<td>377</td>
</tr>
<tr>
<td>Apclid/Fileid List Panel</td>
<td>378</td>
</tr>
<tr>
<td>VSAM DSN Detail - Base Cluster Panel</td>
<td>379</td>
</tr>
<tr>
<td>DSN Detail - Apclid/Fileid List Panel</td>
<td>380</td>
</tr>
<tr>
<td>DSN Detail - User Recovery Exit List Panel</td>
<td>381</td>
</tr>
<tr>
<td>DSN Detail - Comments Panel</td>
<td>382</td>
</tr>
<tr>
<td>DSN Detail - Path List Panel</td>
<td>383</td>
</tr>
<tr>
<td>DSN Detail - Backup List Panel</td>
<td>384</td>
</tr>
<tr>
<td>Delete/Define Statements in ISPF Edit Panel</td>
<td>385</td>
</tr>
<tr>
<td>Archive DSN List Panel</td>
<td>387</td>
</tr>
<tr>
<td>Archive DSN Detail Panel</td>
<td>388</td>
</tr>
<tr>
<td>Additional DSN Archive Data</td>
<td>388</td>
</tr>
<tr>
<td>Backup DSN List Panel</td>
<td>389</td>
</tr>
<tr>
<td>Backup Detail Panel</td>
<td>390</td>
</tr>
<tr>
<td>Backup Detail - Job Information Panel</td>
<td>390</td>
</tr>
<tr>
<td>Log of Logs Selection Panel</td>
<td>391</td>
</tr>
<tr>
<td>Log of Logs Detail Display Panel</td>
<td>392</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>VSAM Sphere Recovery Type Panel</td>
<td>505</td>
</tr>
<tr>
<td>VSAM Sphere Recovery New Names Panel</td>
<td>506</td>
</tr>
<tr>
<td>New Name Pop-Up Panel</td>
<td>507</td>
</tr>
<tr>
<td>SAM Sphere Recovery Panel - Generate JCL (ISPF Edit)</td>
<td>508</td>
</tr>
<tr>
<td>VSAM Sphere Recovery Panel - Problem Warning</td>
<td>509</td>
</tr>
<tr>
<td>VSAM Sphere Recovery - Control Panel - Save JCL</td>
<td>510</td>
</tr>
<tr>
<td>VSAM Sphere Recovery - Control Panel - Submit JCL</td>
<td>511</td>
</tr>
<tr>
<td>Subsystem Detail Panel</td>
<td>555</td>
</tr>
<tr>
<td>Formula for Calculating Repository Size</td>
<td>562</td>
</tr>
<tr>
<td>Purged APPLID/FILEID Reference Report</td>
<td>582</td>
</tr>
<tr>
<td>Purged Backup File Report</td>
<td>583</td>
</tr>
<tr>
<td>Orphaned Backup File Report</td>
<td>584</td>
</tr>
<tr>
<td>Examined Archives Report</td>
<td>584</td>
</tr>
<tr>
<td>Purge Report with Summary-Level Results</td>
<td>586</td>
</tr>
<tr>
<td>Purge Report with Detail-Level Results</td>
<td>587</td>
</tr>
<tr>
<td>Group Setup Example</td>
<td>596</td>
</tr>
<tr>
<td>Archive Accumulation Using the Last Image Process</td>
<td>599</td>
</tr>
<tr>
<td>PTF Full Report</td>
<td>641</td>
</tr>
<tr>
<td>Information Produced with /V Parameter</td>
<td>642</td>
</tr>
<tr>
<td>About System Panel</td>
<td>643</td>
</tr>
<tr>
<td>Sample JCL for RUVZCJC</td>
<td>647</td>
</tr>
<tr>
<td>Sample JCL for RUVZCDA</td>
<td>650</td>
</tr>
<tr>
<td>Sample RUV Data Set Name and Group Definition and Report JCL</td>
<td>652</td>
</tr>
<tr>
<td>ISRSUPC JCL to Search JCL Library</td>
<td>653</td>
</tr>
<tr>
<td>The Outage Free Coordinated Recovery Point Capture Process (Local Site)</td>
<td>671</td>
</tr>
<tr>
<td>Coordinated Recovery Process at the Remote Site</td>
<td>673</td>
</tr>
<tr>
<td>Coordinated Recovery at the Local Site (IMS, DB2, and VSAM)</td>
<td>677</td>
</tr>
<tr>
<td>Coordinated Recovery at the Remote Site (IMS, DB2, and VSAM Involved)</td>
<td>678</td>
</tr>
</tbody>
</table>
Tables

Default DCB Attributes for Output Data Sets ............................................. 80
RUV Command Summary ................................................................. 87
RUV Keywords ................................................................. 92
RUV Keyword Variable Values ...................................................... 105
RUV Subsystem Console Commands ................................................. 123
Journal Model Keywords and Parameters ........................................... 169
Examples of Positional Selection Within a Variable ............................. 174
RUV-Defined Variables ............................................................. 174
RUV Action Variables ............................................................. 179
Rule Set Processing ........................................................................... 199
CICS File Definition Parameters for CICS TS Environments .................. 219
RUVM CICS z/OS Console Commands .............................................. 221
Integration of RUV into an Existing Backup Strategy ............................ 285
Default DCB Attributes for RUV Backup Data Sets .............................. 291
RUV Report Types and Levels .......................................................... 404
RUV Actions for Restore with CURRENT_BACKUP Specified .................. 448
RUV Actions for Forward Recovery .................................................. 467
RUV Actions for Backout Recovery .................................................. 471
Journal Record REJECT Exception Reasons .......................................... 512
Parameter List Structure ................................................................. 520
BMCP Command Summary ........................................................... 539
RUV BCSS Command Summary ....................................................... 544
OUTPUT_MODEL Keywords and Parameters ...................................... 616
About This Book

This book contains detailed information about the Recovery Utility for VSAM (RUV) product. RUV ensures the integrity of your VSAM data around the clock, protecting your online data while providing 24-hour availability of VSAM files. RUV can also journal batch updates to VSAM files. RUV has the ability to track backups and archive data sets so that recovery jobs can be generated automatically.

To use this book, you should be familiar with

- Virtual Storage Access Method (VSAM) concepts and structures
- z/OS systems, Job Control Language (JCL), and the Interactive System Productivity Facility (ISPF)

For example, you should know how to respond to ISPF panels.

How This Book Is Organized

This book is organized as follows. In addition, a glossary of terms and an index appear at the end of the book.

<table>
<thead>
<tr>
<th>Chapter/Appendix Number and Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1, “Introduction to Recovery Utility for VSAM”</td>
<td>Provides a high-level introduction to the product. Summarizes product features and benefits.</td>
</tr>
<tr>
<td>Chapter 2, “Planning for Implementation”</td>
<td>Provides a managerial-level overview of the product and provides guidance for planning to use the product in your environment.</td>
</tr>
<tr>
<td>Chapter 3, “Using RUV Interfaces”</td>
<td>Provides general information for using the RUV batch utilities, ISPF interface, and console commands. Also describes how to set up the RUV default environment.</td>
</tr>
<tr>
<td>Chapter 4, “Using the Batch Journaling Facility”</td>
<td>Describes how to set up and use the Batch Journaling Facility in a batch application environment through RUV rule sets.</td>
</tr>
<tr>
<td>Chapter 5, “Using RUV in a CICS-Only Environment”</td>
<td>Describes how to set up and use RUV in a CICS-only environment. Highlights the benefits of the Backup-While-Open feature.</td>
</tr>
</tbody>
</table>
## How This Book Is Organized

<table>
<thead>
<tr>
<th>Chapter/Appendix Number and Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 6, “Working with Archives”</td>
<td>Describes how to manage and use RUV archives (which contain change records for recovery).</td>
</tr>
<tr>
<td>Chapter 7, “Working with Backups”</td>
<td>Describes how to use RUV for backup processing. Also covers backups performed by non-RUV processes.</td>
</tr>
<tr>
<td>Chapter 8, “Creating Reports”</td>
<td>Describes how to obtain information from the RUV repository.</td>
</tr>
<tr>
<td>Chapter 9, “Reorganizing VSAM Files”</td>
<td>Describes how to use RUV for reorganizing VSAM files.</td>
</tr>
<tr>
<td>Chapter 10, “Restoring Data Sets”</td>
<td>Describes how to use RUV for restore processing. Also covers restores performed by non-RUV processes.</td>
</tr>
<tr>
<td>Chapter 11, “Recovering VSAM Data Sets”</td>
<td>Describes how to use RUV for general recovery of VSAM files. Provides detailed information about forward and backout recovery. Describes how to use the Recovery user exit.</td>
</tr>
<tr>
<td>Chapter 12, “Using RUV for Disaster Recovery”</td>
<td>Describes how to use RUV for disaster recovery and other recovery purposes in a remote environment.</td>
</tr>
<tr>
<td>Chapter 13, “Working with BMC Software Subsystems”</td>
<td>Provides details about the BMC Software subsystems and describes commands for working with them.</td>
</tr>
<tr>
<td>Chapter 14, “Maintaining the RUV Repository”</td>
<td>Provides details about the RUV repository and describes utilities for maintaining the repository.</td>
</tr>
<tr>
<td>Chapter 15, “Using VSAM Groups”</td>
<td>Describes how to define, use, and manage RUV groups of VSAM files.</td>
</tr>
<tr>
<td>Chapter 16, “Using Sequential Groups”</td>
<td>Describes how to define, use, and manage RUV groups of sequential files.</td>
</tr>
<tr>
<td>Appendix A, “Solving Problems”</td>
<td>Describes how to contact Product Support and how to collect diagnostic information.</td>
</tr>
<tr>
<td>Appendix B, “Migrating from RPCV to RUV”</td>
<td>Provides information for conversion from the BMC Software RECOVERY PLUS for CICS/VSAM product.</td>
</tr>
<tr>
<td>Appendix C, “Information Gathering Forms”</td>
<td>Provides blank forms you can photocopy and use for gathering information about your environment. You can use this information while planning for and implementing RUV.</td>
</tr>
<tr>
<td>Appendix D, “A Coordinated Recovery Process”</td>
<td>Describes how to use RUV with other BMC Software recovery management products to handle coordinated recoveries across IMS, DB2, and VSAM environments.</td>
</tr>
<tr>
<td>Appendix E, “Reference Summary”</td>
<td>Provides summary information about RUV commands, keywords, values, variables, and other topics.</td>
</tr>
</tbody>
</table>
Related Documentation

This product is supported by several types of documentation:

- online books
- online Help
- release notes and other notices

The following publications provide information about RUV:

- *Recovery Utility for VSAM Installation and Customization Guide*
- *Recovery Utility for VSAM User Guide*
- *Recovery Utility for VSAM Messages Manual*

The following BMC Software publications provide information about the SNAPSHOT UPGRADE FEATURE (SUF) component that is used for Instant Snapshot copy processing:

- *EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE General Information*
- *EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE Installation and Customization Guide*
- *EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide*

These publications are available for download from the BMC Software EPD site and are included on the RUV documentation CD.
Online Books

The books that accompany this product is available in online format and printed format. If you are a Windows or Unix user, you can view online books with Acrobat Reader from Adobe Systems. The reader is provided at no cost.

Online books are formatted as Portable Document Format (PDF) files. You can view them, print them, or copy them to your computer by using Acrobat Reader 3.0 or later. You can access online books from the documentation compact disc (CD) that accompanies your product or from the World Wide Web.


Online Help

The RUV product includes online Help. In the RUV ISPF interface, you can access Help by pressing F1 from any ISPF panel.
# Conventions

This section provides examples of the conventions used in this book and explains how to read syntax statements.

## General Conventions

This book uses the following general conventions:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>information that you are instructed to type</td>
<td>Type <code>SEARCH DB</code> in the designated field. Type <code>search db</code> in the designated field. (Unix)</td>
</tr>
<tr>
<td>specific (standard) keyboard key names</td>
<td>Press <code>Enter</code>.</td>
</tr>
<tr>
<td>field names, text on a panel</td>
<td>Type the appropriate entry in the Command field.</td>
</tr>
<tr>
<td>nonspecific key names, option names</td>
<td>Use the HELP function key. Use the <code>KEEPDICTIONARY</code> option.</td>
</tr>
<tr>
<td>z/OS calls, commands, control statements, keywords, parameters, reserved words</td>
<td>Use the SEARCH command to find a particular object. The product generates the RECOVER FORWARD statement next.</td>
</tr>
<tr>
<td>code examples, syntax statements, system messages, screen text</td>
<td><code>//STEPLIB DD</code> The dsn <code>dsname</code> is not available.</td>
</tr>
<tr>
<td>emphasized words, new terms, variables</td>
<td>The instructions that you give to the software are called <code>commands</code>. In this message, the variable <code>file_name</code> represents the file that caused the error.</td>
</tr>
<tr>
<td>single-step procedures</td>
<td>» To enable incremental backups, type <code>y</code> and press <code>Enter</code> at the next prompt.</td>
</tr>
</tbody>
</table>
This book uses the following types of special text:

---

**NOTE**

Notes contain important information that you should consider.

---

**WARNING**

Warnings alert you to situations that could cause problems, such as loss of data, if you do not follow instructions carefully.
Syntax Statements

Syntax statements appear in Courier. The following example shows a sample syntax statement:

```
COMMAND KEYWORD1 [KEYWORD2|KEYWORD3] KEYWORD4={YES|NO}  
file_name...
```

The following table explains conventions for syntax statements and provides examples:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items in italic type represent variables that you must replace with</td>
<td>BACKUP_IN(dsn44)</td>
</tr>
<tr>
<td>a name or value.</td>
<td></td>
</tr>
<tr>
<td>Brackets [ ] indicate a group of options; one of the options is</td>
<td>[ ALL</td>
</tr>
<tr>
<td>required.</td>
<td></td>
</tr>
<tr>
<td>Braces { } indicate a group of options. You can choose at least one</td>
<td>{ LOCATION(locname8) }</td>
</tr>
<tr>
<td>of the items in the group, but none of them is required. Do not</td>
<td>{ COMMENT(comment_fields) }</td>
</tr>
<tr>
<td>type the braces when you enter the option. A comma means that you</td>
<td>{ UNIT_OF_WORK_IN ( ddn8_or_dsn44 { , ddn8_or_dsn44, ... }</td>
</tr>
<tr>
<td>can choose one or more of the listed options. You must use a comma</td>
<td></td>
</tr>
<tr>
<td>to separate the options if you choose more than one option.</td>
<td></td>
</tr>
<tr>
<td>An asterisk * indicates a wildcard. A question mark ? indicates a</td>
<td>dsn??,*,...</td>
</tr>
<tr>
<td>wildcard substitution for one character. Three ellipsis indicates a</td>
<td></td>
</tr>
<tr>
<td>repetition of the listing.</td>
<td></td>
</tr>
<tr>
<td>Commands in bold specify required items. You must enter the items</td>
<td>BACKUP REPOSITORY</td>
</tr>
<tr>
<td>as part of the overall command statement. Do not type the bold</td>
<td></td>
</tr>
<tr>
<td>when you enter the item.</td>
<td></td>
</tr>
<tr>
<td>A vertical bar means that you can choose only one of the listed</td>
<td>(ACTIVE</td>
</tr>
<tr>
<td>items. In the example, you would choose either ACTIVE or INACTIVE.</td>
<td></td>
</tr>
<tr>
<td>An ellipsis indicates that you can repeat the previous item or items</td>
<td>ddn8, . . .</td>
</tr>
<tr>
<td>as many times as necessary.</td>
<td></td>
</tr>
<tr>
<td>Default values are listed first. An underline indicates a default.</td>
<td>KEYWORD4(YES</td>
</tr>
<tr>
<td>In the example, YES is the default. Do not type the underscore.</td>
<td></td>
</tr>
</tbody>
</table>
References to RUV Libraries

In this document, the term RUV sample library refers to the library that was distributed with the name RUV.CNTL. The term RUV load library refers to the library that was distributed with the name RUV.LOAD. Typically, these libraries are renamed during the installation process with a high-level qualifier that is valid in your environment.

References to CICS

Any reference to the term CICS refers to all IBM-supported versions and releases of the CICS Transaction Server product. The specific product name, version, and release numbers are noted only when this information is significant.

Support for CICS Journals

**NOTE**

As stated in the Release Notes, RUV only supports CICS releases that are currently supported by IBM as at the time of this publication.

The older CICS journal infrastructure is no longer supported by IBM, since the journalling function is entirely implemented with log streams for current IBM-supported releases of CICS. References in this manual to the older journal infrastructure (e.g. journal extents) and related RUV commands are retained for compatibility only.

While it might be possible for you to successfully process older type (i.e. non-log stream) journals or log streams generated by unsupported CICS releases (depending on your environment), we do not provide support or maintenance for this type of processing or guarantee that it will continue to function successfully in future releases of RUV or after maintenance updates.
Introduction to Recovery Utility for VSAM

This chapter introduces the RECOVERY UTILITY for VSAM (RUV) product. The following topics are included:

- Recovery of Data Sets with RUV ........................................ 39
  - Data Set Structures ............................................. 40
  - Recovery Process ................................................ 41
- Recovery Operations .................................................. 42
  - Backout Recovery Operation .................................... 42
  - Forward Recovery Operation ................................... 42
  - Unit-of-Work Recovery Operation ............................... 43
- Types of Backups Needed for a Recovery .......................... 44
  - Crisp Backup ...................................................... 44
  - Fuzzy Backup ..................................................... 45
  - Recognized Backup Formats ..................................... 45
- Recovery Process in RUV ............................................. 46
- RUV Features .......................................................... 47
  - Unit-Of-Work Recovery .......................................... 47
  - Definition and Creation Of Groups .............................. 48
  - Last Image Accumulation of VSAM File Archives ............. 48
  - Support of IAM Disk Management of VSAM Files .......... 48
  - RBA0 Detection Support .......................................... 49
  - Automatic Tracking of Recovery Resources ................... 49
  - Backup of VSAM Data Sets ...................................... 49
  - Instant Snapshot Copies of VSAM Data Sets .................. 50
  - Archiving of CICS Journal Data .................................. 50
  - Journaling of VSAM Data Sets Updated in Batch Mode ...... 50
  - Automatic Step Backout for Batch Jobs ......................... 51
  - Forward and Backout Recovery of VSAM Data Sets .......... 51
  - Automated Management of RUV Archives and Backups ....... 52
  - Extensive Reporting and Selection Capability ................. 52
  - Subsystem Technology ............................................ 52
  - Repository Recovery with SMF Data Sets ...................... 52
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Considerations for Using RUV</td>
<td>53</td>
</tr>
<tr>
<td>VSAM Data Set Types Not Supported</td>
<td>53</td>
</tr>
<tr>
<td>Compressed ESDS</td>
<td>53</td>
</tr>
<tr>
<td>Size of VSAM Records</td>
<td>54</td>
</tr>
<tr>
<td>Parallel Processes</td>
<td>54</td>
</tr>
</tbody>
</table>
Recovery of Data Sets with RUV

Recovery processing after a data loss can mean the difference between operating at maximum efficiency and not operating at all. For users of CICS, VSAM data loss creates a need to recover large amounts of mission-critical data. The process of recovering CICS VSAM data is complex, and few qualified products exist to accomplish the task.

The Recovery Utility for VSAM (RUV) product provides quick and easy VSAM data recovery processing. RUV allows you to focus on the process of recovering data rather than the program that recovers the data.

You can use RUV to recover VSAM data sets in the following situations:

- Physical VSAM data is lost or damaged, which may occur for the following reasons:
  - A hardware failure occurred.
  - A part of a DASD is not readable.
  - The data was deleted.
  - A natural disaster, such as fire or storm, occurred.

- Logical data integrity is lost, which may occur for the following reasons:
  - An application error occurred.
  - A failure occurred during CICS transaction backout.

Loss of logical data integrity occurs when a CICS application incorrectly updates your VSAM spheres. CICS transaction backout maintains logical data integrity by backing out partially completed transactions from VSAM data sets. Transaction backouts can fail under some conditions. For example, an I/O error can occur when CICS accesses the VSAM data set. In this type of transaction backout failure, RUV recovers your damaged VSAM data sets and all updates that were made before the problem occurred.

The recovery process is a sequence of tasks that begins with the last usable backup, and may include various logs, journals, and archival data sets. The recovery process is complete when data is recovered and processing capability is restored. Figure 1 on page 40 illustrates how the recovery process ensures business continuity of the data processing environment.
As shown in Figure 1, actual loss of data may have occurred during a very short span of time. The time needed to completely recover data sets, however, which includes the elapsed disruption time, is a long span of time.

RUV recovers VSAM data sets in a forward mode or a backout mode.

**Data Set Structures**

RUV processes the following VSAM data set structures and organizations:

- VSAM key-sequenced data sets (KSDS)
- VSAM entry-sequenced data sets (ESDS)
- VSAM fixed and variable length relative-record data sets (RRDS)

The various data set structures are generically referred to as VSAM spheres. A VSAM sphere consists of the base cluster and any alternate index paths that are associated with the base cluster. RUV identifies a VSAM sphere by the name of the base cluster.
Recovery Process

Recovery of VSAM data sets is an automated process within the RUV environment. RUV collects and records recovery information; however, it is your responsibility to determine when recovery processing is needed and which data sets require recovery.

As shown in Figure 2, RUV can gather and coordinate information from many sources, including z/OS log streams, CICS journals, RPCV journals, RUV batch journals, and RUV backups. The recovery information is recorded in the RUV repository. With this recorded information, recovery is simply a matter of selecting the VSAM data sets to be recovered. RUV automatically selects which backups and archives are needed and constructs the correct recovery JCL.

Figure 2  RUV System Flow
Recovery Operations

This section explains the types of recovery that you can perform with RUV.

Backout Recovery Operation

You can perform a backout recovery operation to roll back a data set to an earlier state. For example, when an application program updates a data set incorrectly or when an abend causes an incomplete update, the data set must be recovered to its previously known condition. The backout recovery operation uses the current copy of a data set and all before-image journal records that were created since the point-in-time specified for recovery (see Figure 3). See “Coding the RECOVER BACKOUT Statements” on page 490 for the command syntax.

Forward Recovery Operation

After a data set becomes physically damaged, you can perform a forward recovery operation to update the data set from an earlier state to the current time or a specified time (see Figure 4). See “Coding RECOVER FORWARD Statements” on page 474 for the command syntax.
Unit-of-Work Recovery Operation

A unit-of-work (UOW) recovery operation is a forward recovery that applies only completed transactions to the data set (see Figure 5). The UOW recovery operation uses a backup of a data set and all after-image journal records that were created from the time the backup was made until the point in time specified for recovery. Updates that are part of an incomplete UOW are not applied to the current recovery and can be written to a separate output file. You can view this file to identify which transaction data was not applied to the recovery. The output data can also be used as input for a subsequent UOW recovery. See “Coding the RECOVER COMPLETED_UNIT_OF_WORK Statements” on page 484 for the command syntax.

Figure 5 Unit Of Work Recovery Process
Types of Backups Needed for a Recovery

Before performing a recovery, you must verify that all the data has been backed up or archived and registered in the RUV repository. This section describes the types of backups and how RUV uses the different backups for a good recovery.

In the simplest terms, before you can perform a recovery you must have a valid backup of the VSAM files and all the changes that were made to the file since the backup was performed. This section assumes that the backup and recovery are performed on the local system; however, the principles also apply to remote recovery and disaster recovery operations.

Crisp Backup

In an ideal backup situation (which is rare), all transactions are paused while a backup is made. Figure 6 illustrates this ideal situation. Prior to a backup point, multiple transactions (T1, T2, and so on) are processed by CICS, and the DFHLOG collects the log stream information about each transaction. At the backup point, all transaction activity pauses while a backup is made. When the backup is completed, the processing of transactions resumes. Few organizations can afford to bring their operations to a stop while they do a crisp backup.

Figure 6 Crisp Backup
Fuzzy Backup

A more realistic backup situation is illustrated in Figure 7. During the time that the backup is being made, CICS continues processing transactions. This fuzzy backup contains both completed transactions (T1 and T2) and partially completed (in-flight) transactions (T3 and T4). During the time of the backup, the log stream (DFHLOG) continues to record all transaction activities in a time sequence.

Figure 7  Fuzzy Backup

RUV allows you to specify a fuzzy backup using the Backup-While-Open (BWO) feature. This is especially important to operations that require system and application availability 24 hours a day, 7 days a week. A recovery must be run against the data set when a fuzzy backup is restored.

Recognized Backup Formats

RUV can generate fuzzy backups of VSAM data sets and use these backups and archives for recovery. RUV also recognizes SMS backups. During a recovery, RUV automatically checks for SMS backups and uses the more current SMS or RUV backup.

RUV also supports alternate backup techniques and can restore the resulting backup files (called external backups) automatically in a recovery situation. To use external backups, you must provide required information in a one-time definition process.
Recovery Process in RUV

To perform an RUV forward recovery, you must have a valid backup of the VSAM data and the archive and the files must be registered in the RUV repository.

The RUV recovery process is illustrated in Figure 8.

**Figure 8  Recovery from a Backup**

When you specify a recovery point, RUV determines which backup (or backups) and archive (or archives) to use.

- If the recovery point corresponds to the time of a crisp backup, RUV simply restores the backup.

- If the backup was a fuzzy backup, RUV restores the backup and applies the transactions (journal data) that occurred from the start of the backup until a point in time after the completion of the backup. With the combination of the fuzzy backup and the journal data, the integrity of the data is maintained.
**RUV Features**

RUV offers the following features:

- unit-of-work recovery
- definition and creation of groups
- last image accumulation of VSAM file archives
- support of IAM disk management of VSAM files
- RBA0 detection support
- automatic tracking of recovery resources
- backup of VSAM data sets (including Backup-While-Open to CICS)
- Instant Snapshot copies of VSAM data sets
- archiving of CICS journal data
- journaling of VSAM data sets updated in batch mode
- automatic step backout for batch jobs
- forward and backout recovery of VSAM data sets
- automated management of RUV archives and backups
- extensive selection and reporting capability
- use of subsystem technology (for error detection and I/O interception)
- flat file support

**Unit-Of-Work Recovery**

RUV supports unit-of-work (UOW) recovery. A UOW recovery is a forward recovery that applies only completed transactions. The UOW recovery uses a backup of a data set and all after-image journal records that were created from the time the backup was made until the point in time specified for recovery. Updates that are part of an incomplete UOW are not applied to the current recovery and can be written to a separate output file. You can view this file to identify the transaction data that was not applied to the recovery. The output data can also be used as input for a subsequent UOW recovery.

To prepare the environment for a UOW recovery, you must configure the system to log specific information about the transactions and use the ARCHIVE command with the UNIT_OF_WORK_OUT keyword. You must also perform an initial backup to provide a known starting point for future UOW recoveries.
Definition and Creation Of Groups

The RUV Group Feature provides a flexible method for working with a collection of RUV-registered files in much the same way that you work with a single file. After the files have been registered and a group has been defined, the VSAM_GROUP or SEQ_GROUP keyword can replace the VSAM_FILE or SEQ_FILE keyword in most commands, including BACKUP, RESTORE, RECOVER, and ARCHIVE.

For more information about VSAM groups, see Chapter 15, “Using VSAM Groups.”

Last Image Accumulation of VSAM File Archives

RUV provides a LAST_IMAGE feature that allows a single after-image record to be retained for forward recovery purposes. When RUV is accumulating archives with the LAST_IMAGE or COPY keyword, RUV sorts all data and then discards all the after-image records except the most current records.

By accumulating last image archives, the fewest number of records are retained. This selective processing provides a quick forward recovery because only the minimum number of records are processed to reconstruct a VSAM file.

Support of IAM Disk Management of VSAM Files

IAM is a reliable high performance disk file manager that can be used in place of VSAM KSDS or VSAM ESDS data sets for batch, TSO, and online processing.

RUV provides complete support for IAM key-sequenced data set (KSDS) files, entry-sequenced data set (ESDS) files, relative-record data set (RRDS) files, and ESDS and KSDS files with alternate indexes (AIX), also known as paths.

RUV does not back out changes to any IAM ESDS file with or without paths. Only forward recovery can be performed on IAM ESDS files.

Before running RUV backup, restore or batch processing, the IAM product must be initialized.
RBA0 Detection Support

Through the RBA0 backup detection feature, RUV can detect that a VSAM file is empty and register an RBA0 backup at that point in time. It can also register an RBA0 backup when a file that has been defined with the REUSE option is opened for output. (This action causes the high used relative byte address to be reset to zero, which effectively deletes all records in the file.)

An RBA0 backup is simply a record in the repository; no physical data set is associated with an RBA0 backup. However, an RBA0 backup is intended to be used, in the same way as any other type of backup, as a starting point for recovery. When an RBA0 backup is selected during restore or recovery processing, it causes a delete/define operation for the VSAM file.

Automatic Tracking of Recovery Resources

RUV records recovery information in the RUV repository. The RUV repository (also known as a REGISET) is a VSAM data set (allocated with SHAREOPTIONS=4). Multiple CPU access to the repository is allowed because RUV uses enqueues and DASD reserves to protect the repository from unauthorized access and update. The data within the repository is further protected by providing for multiple copies of the repository. RUV supports up to 11 primaries. Each primary can be duplexed. Primaries and duplexes are automatically cross-checked for validation. For more information on the RUV repository, see Chapter 14, “Maintaining the RUV Repository.”

Backup of VSAM Data Sets

Creation of a complete backup of the file to be recovered is absolutely essential for successful file recovery. RUV provides fast backup of KSDS, ESDS, RRDS, and VRRDS VSAM spheres. RUV records enough information during a sphere backup to allow it to automatically delete/define the file and rebuild any alternate indexes during a restore. RUV can also create flat file backups identical to those created with the IDCAMS utility REPRO command.

The Backup-While-Open (BWO) feature of RUV is especially important to CICS users who need 24 by 7 support. RUV can back up a VSAM data set even as it is being updated by a CICS region. The result is a fuzzy (probably complete) backup. Recovery must be run against the data set when a fuzzy backup is restored.

Unless you specify a specific backup name, all backups restored by RUV are the most current backups available. You can enable or disable the use of backups according to their type and the process that RUV is performing.
Instant Snapshot Copies of VSAM Data Sets

RUV uses the BMC Software SNAPSHOT UPGRADE FEATURE (SUF) for VSAM component to create Instant Snapshot copies of VSAM data sets. An Instant Snapshot copy is a duplicate of a data set that can be created in seconds by exploiting the technology that is available with intelligent storage devices. The Instant Snapshot copy process is nearly instantaneous, regardless of the size of the data set. An Instant Snapshot copy can also be restored in seconds. RUV automatically recognizes and handles the Instant Snapshot copy during the restore process.

To create Instant Snapshot copies, you must install, customize, and implement SUF. This component is packaged for distribution with the BMC Software EXTENDED BUFFER MANAGER (XBM) product. SUF for VSAM is included with RUV at no additional charge. SUF requires an active XBM subsystem for creating Instant Snapshot copies and for performing an instant restore from an Instant Snapshot copy.

You also need supported hardware and software from your storage device vendor. For more information, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

Archiving of CICS Journal Data

RUV reads recovery information from CICS 4.x journal extents and Transaction Server log streams. It records the recovery information to one or more RUV archives. RUV registers all output archives in the RUV repository.

RUV automatically tracks the relationship between the CICS APPLID/FILEID and the VSAM file data set name (DSN). RUV can use Transaction Server’s log of logs to quickly build an inventory of VSAM files that are being updated by CICS regions.

The batch journal (archive) data set and the archive data set that RUV produces from CICS extents have the same format. Identical formats simplify recovery planning.

Journaling of VSAM Data Sets Updated in Batch Mode

RUV provides batch programs with the same journaling capabilities and protection that CICS transactions have. You decide which batch jobs, programs, and data sets that RUV will protect and where RUV will write the recovery information; RUV does the rest of the work. RUV can capture before-images, after-images, or both. RUV uses before-images to back out changes to a VSAM data set; RUV uses after-images for forward recovery.
All recovery images for a single job step are written to a log file, which is referred to as an RUV archive. You do not need to change existing JCL to use the Batch Journaling feature. The Batch Journaling feature may reduce the number of backups necessary for a VSAM file. For more information, see Chapter 4, “Using the Batch Journaling Facility.”

Automatic Step Backout for Batch Jobs

RUV can record before-images to a log file. RUV can then use the before-images to remove the changes made to VSAM files by a batch program when a job step abends. You decide which batch jobs and programs RUV will protect and where RUV will write the log. You do not need to change existing JCL to use step backout features.

Forward and Backout Recovery of VSAM Data Sets

RUV can perform either forward or backout recovery of a damaged data set. Forward recovery is required if a data set has been physically damaged. Forward recovery may be used instead of a backout if before-images were not recorded for the data set being recovered. Forward recovery requires a backup to be available. Typically, backout recovery is used if a job step has abended or the data has recently been logically damaged. Backout recovery is frequently faster than forward recovery because typically less data is needed.

At a minimum you must determine

- when a recovery needs to be run
- which file or files need to be recovered
- the type of recovery (forward or backout)

You can use the ISPF interface to generate the recovery JCL, as described in “Using the ISPF Interface for Recovery” on page 495. Or you can use the flexible command syntax and automatic selection features of the batch interface to create the recovery JCL manually, as described in “Using RUV for Forward Recovery” on page 473.

You may request an inventory of backups and archives that will be needed for recovery processing.

You do not need to be concerned about the origin of the recovery data. RUV uses recovery information from both CICS and RUV batch archives to recover VSAM spheres. RUV exploits multi-tasking and multiple address spaces during the recovery process to provide the fastest recovery possible.
Automated Management of RUV Archives and Backups

RUV backups and archives are registered automatically in the repository. RUV can maintain multiple copies of backups and multiple copies of archives. You can create special archives for selective processing with the Accum function. This function allows you to gather recovery information for a file or group of files in a single archive. The single archive allows you to easily manage the recording of archive information for local or off-site usage. In addition, accumulating archives speeds recoveries by reducing the number of records to be passed during recovery processing.

You can manage obsolete backups and archives easily with the purge process. You can specify when a backup file should be considered obsolete by either number of days of retention or number of backups. RUV automatically handles expiration of archives based on the status of the backups related to an archive. If you are not using RUV backups, archives may be purged based on number of days of retention. The purge process removes obsolete recovery information from the repository and performs DASD and tape cleanup for z/OS by allowing the obsolete files to be eligible for scratching and deleting, as necessary.

Extensive Reporting and Selection Capability

RUV offers extensive reporting capability on information in the repository. Many of the reporting fields support wildcard selection, and numerous fields offer filtering to tailor the output to your needs. Summary, detail, and full reporting levels are available for most reports.

Subsystem Technology

RUV employs z/OS subsystem technology to detect the opening of VSAM files. For ADD, UPDATE, or DELETE actions, if you have requested batch journaling for the file, RUV intercepts update activity on the file and journals the updates. RUV uses subsystem technology to detect the end of the step and to determine whether the step has abended. The z/OS subsystem manages I/O to the repository and performs APF-authorized functions.

Repository Recovery with SMF Data Sets

RUV can record all changes to the repository by writing recovery information to z/OS system management facilities (SMF) data sets.
Sarbanes-Oxley (SOX) reporting

RUV provides a REPORT AUDIT command to extract Archive (change) data by transaction, file ID, terminal ID, and time range.

Considerations for Using RUV

This section discusses issues you should be aware of when using RUV.

VSAM Data Set Types Not Supported

RUV does not support the following types of VSAM data sets:

- native update of AIX data sets
- spanned ESDSs
- linear data sets
- control interval (CI) mode update data sets
- backout recovery for extended ESDSs
  (forward recovery for these data sets is supported)
- extended RRDSs
- IAM files with paths (IAM version 7.0 or higher)

RUV does not perform recovery for ESDSs that are accessed through a path.

RUV does not support backout recovery for extended format ESDSs. Forward recovery for these data sets is supported.

RUV does not perform backout recovery for multiple-volume ESDSs that have paths. To obtain the same type of recovery results for this type of data set, you can perform a forward recovery to a point in time.

Compressed ESDS

RUV uses a relative byte address (RBA) to identify records in a VSAM ESDS. Compression products such as DATA ACCELERATOR may change the RBA of an ESDS record as updates occur. Because the RBAs of compressed ESDS are unpredictable, these data sets should be excluded from compression if they are to be recovered using RUV.
LIKE Allocation

RUUV does not journal VSAM files that are defined and allocated through the LIKE parameter on the DD statement. The new data set is not cataloged until the end of the step; consequently, it is treated as a temporary data set.

Size of VSAM Records

For recovery processes, RUV supports a maximum VSAM record size of 32 KB. For backup and restore processes, RUV supports a maximum VSAM record size of 64 KB.

Parallel Processes

RUUV provides parallel subtask processing as it reads input files, such as backup or archive files. This subtask processing allows input/output (I/O) overlapping among the various tasks.

RUUV uses up to 100 task control blocks (TCBs) (also called reader tasks) to achieve this performance. You can specify the maximum number of reader tasks through an optional setting, SET DEFAULT READER_TASKS(nn). The number of tasks that RUV will manage, however, is determined by the memory resources available in your environment. For example, during a recovery process, RUV attaches SORT. The system resources provided to SORT can be site specific, but also be unknown to RUV.

For further information on using reader tasks, see “Using the READER_TASKS Keyword” on page 134.

RUUV performance can be impacted by the number of buffers provided by or allowed for each QSAM open. RUV does not manipulate the number of buffers acquired. Other vendor products may cause this number of buffers to increase at execution time.
Planning for Implementation

This chapter describes how to plan for implementation of RUV. It provides a high-level overview of the RUV concepts and procedures for implementing RUV. For task-oriented information, see Chapters 3 through 13.

The following topics are included in this chapter:

- Overview of the Batch Journaling Facility .......................................................... 56
- Planning to Use the Batch Journaling Facility ...................................................... 57
  - Identifying VSAM Files to Enable for RUV Recovery ...................................... 57
  - Identifying Jobs and Programs That Update VSAM Data Sets ......................... 59
- Determining Recovery Actions ........................................................................... 59
- Defining Journal and Log Models ....................................................................... 60
- Creating User Variables .................................................................................... 61
- Organizing and Coding Rule Sets ...................................................................... 62
- Merging and Testing Rule Sets .......................................................................... 63
- Reducing the Number of Backups ...................................................................... 64
- Planning for Archive Management .................................................................... 65
  - Goals of Archive Management ......................................................................... 65
  - How RUV Provides CICS-Like Processing .................................................... 65
  - Collecting and Managing Archives .................................................................. 66
  - Accumulating Archives .................................................................................... 68
  - Copying or Moving Archives .......................................................................... 69
  - Maintaining the Repository ............................................................................. 69
- Managing Backups .............................................................................................. 70
- Restoring Data Set Backups .............................................................................. 71
- Obtaining RUV Reports ..................................................................................... 72
- Using RUV for Local Recovery .......................................................................... 73
- Using RUV for Remote Recovery ...................................................................... 74
- Using RUV in CICS-Only Environments ............................................................ 75
  - CICS Requirements ......................................................................................... 75
  - Journal and Log Stream Archiving .................................................................. 75
  - Backup-While-Open Feature ........................................................................... 76
  - Repository Access ............................................................................................ 76
  - Recovery ........................................................................................................... 76
Overview of the Batch Journaling Facility

This section explains the major Batch Journaling Facility functions and the planning steps necessary to use them.

**NOTE**
If you have a CICS-only environment, you do not need to use the Batch Journaling Facility.

RUV is a highly automated utility that performs sophisticated recovery operations quickly and thoroughly. Some users will establish RUV recovery operations using the online interface. But for other users, recovery processing is time consuming, complicated, and dependent on batch windows. These organizations can use the batch interface to set up and control recovery for large numbers of VSAM files.

The RUV Batch Journaling Facility (see Figure 9) allows you complete control of VSAM data set journaling. You analyze data sets and determine which ones need recovery protection. These data sets can be recovered individually, in total, or in user-defined groups. After you make this determination, you use RUV rule sets to specify how a particular recovery scheme will work in your environment.

**Figure 9  Batch Journaling Facility Process**

The RUV Subsystem and the BMC Software Primary Subsystem are required for most RUV processing features. (Chapter 13, “Working with BMC Software Subsystems,” describes the BMC Primary Subsystem.)
Planning to Use the Batch Journaling Facility

This section describes the planning steps that are required for successful implementation of the Batch Journaling Facility. Chapter 4, “Using the Batch Journaling Facility,” discusses rule sets and the commands that you can use to create them. Appendix E, “Reference Summary,” provides detailed reference information for all RUV commands, keywords, values, and variables.

The description of the following planning steps will assist you in implementing the Batch Journaling Facility:

1. Identify the VSAM files to enable for RUV recovery.

2. Identify the jobs and programs that update or modify the RUV-recoverable VSAM data sets.

3. Determine the recovery actions to take if a program failure occurs.

4. Define journal and log models.

5. Create user variables for use in constructing data set names and JCL.

6. Organize and code RUV rule sets. The RUV.CNTL library contains completed rule set samples.

7. Merge and test rule sets.

8. Reduce the number of backups.

Chapter C, “Information Gathering Forms,” provides sample forms to help you collect the planning information that is discussed in this section. The RUV.CNTL library contains completed rule set samples.

Identifying VSAM Files to Enable for RUV Recovery

The first step in implementing the Batch Journaling Facility is to decide which VSAM files you want to enable or not want to enable for RUV recovery. To enable a file for recovery, you set up a definition that

- tells RUV to create an archive (journal or log) of the changes made to the file
- provides other information necessary for archiving and recovery tasks.
Sources of Information

Examining your CICS file definitions is a good way to start because many VSAM files are updated by both CICS and batch programs. Use the TSO Data Set List utility (usually defined as ISPF main menu option 3.4) to help identify other VSAM files.

Data Set Names

RUV identifies files by using the data set name (DSN) of the base cluster. RUV file selection allows you to use wildcard characters to select sets of files easily. Use of wildcard characters also eases the maintenance of your system. You will have fewer specifications to code, and new files can be added to your system without requiring maintenance to RUV.

Types of Recovery

As you identify data sets that are candidates for recovery, decide whether you need forward recovery, backout recovery, or both.

- You would choose backout recovery to allow for a rapid recovery if a job step abend occurs. RUV uses its log to back out changes to recoverable VSAM files. This operation is typically faster than deleting and defining the VSAM file, restoring the entire VSAM file, and rerunning one or more jobs.

- You would choose forward recovery to allow for a rapid recovery if a hardware or logic error invalidates a file and you must return the file to the condition that it was in before the error occurred. Forward recovery is commonly used at disaster recovery sites.

Data Set Exclusion

It is important to decide which files you do not want to enable for recovery. Unnecessary recoverability increases the run-time for jobs and requires more DASD resources for logs and journals. Examples of data sets that you do not want to enable for recovery are IMS RECON data sets, data sets used by Netview or MQ Series, and VSAM data sets used by z/OS subsystems. You will have fewer specifications to code if you use wildcard characters for generic file exclusion.
Identifying Jobs and Programs That Update VSAM Data Sets

The second implementation step is to identify jobs and programs that update or modify the recoverable VSAM data sets. Good sources of information are internal documentation and SMF reports. If you use wildcard characters for generic job and program inclusion and exclusion, you will have fewer specifications to code and manage.

Job Exclusion

Many utility programs should be excluded from logging or journaling. These programs include IDCAMS, DFHSIP, HSM, and SMP/E, as well as many others. Many of these utilities have been identified and are listed in a sample exclusion list in the RUV.CNTL library.

Job Step Backout

As you identify jobs and programs that use protected VSAM data sets, decide whether to use step backout recovery for a particular program. Performing step backout recovery is typically faster than restoring files and performing a forward recovery.

Determining Recovery Actions

The third implementation step is to determine which recovery actions to take if a program failure or VSAM data set failure occurs.

No Action for Program Failure

RUV can provide recovery protection and still perform no special actions if a program failure occurs. Your organization would perform its standard error recovery procedures and then use RUV to back out step changes. This setup is usually the best choice for test jobs and may also be appropriate for production jobs.
Creation of a Step Backout Job

RUV can easily build a step backout job for you. It can either submit the job to JES for immediate execution or store it in a PDS for manual submission at a later time. You must supply skeleton JCL for the backout job or jobs as required by your system. Additionally you must provide RUV with information about the name of the PDS or the JES submission classes to use. A sample recovery job is provided in the RUV.CNTL library that works well on most systems. Refer to member $INDEX in the RUV.CNTL library for a list of sample jobstreams. The use of step backout is usually a good choice for production jobs.

Journal Failure

You should decide what actions you want RUV to take if a journal failure occurs. RUV offers you the following options:

- Abend the job immediately.
  
  Typically you can back out changes, correct the problem with the journal, and rerun the step.

- Continue running the job.
  
  You would need to back up unprotected files after the job has completed. Typically, journal failures are caused by lack of space. Determine whether you need to adjust journal allocation parameters.

Defining Journal and Log Models

The fourth implementation step is to define journal models and log models. Journal and log models act as templates for constructing a DD allocation statement for QSAM logs and journals. RUV records recovery information to these QSAM files. This section discusses the decisions you must make to successfully set up a logging and journaling environment. Chapter 4, “Using the Batch Journaling Facility,” provides instructions for creating journal and logging models.

Contents and Disposition of Logs and Journals

RUV logs contain step backout information and are typically deleted when a job step successfully completes. If a step abends, the log is retained and used as input for RUV step backout.
RUV journals can contain backout information as well as forward recovery information; however, they are intended for long term storage of recovery data. Journals are retained regardless of the step completion code. Journals are typically removed from the system by purge processing (an RUV maintenance job).

Size of Logs and Journals

The log or journal must be able to hold all of the updates that are being performed in any single job step, but precise size control is not required. Unused DASD space can be automatically released, and RUV allocates additional logs or journals if you have requested X37 support.

Data Set Names of Logs and Journals

Each allocated log or journal must have a unique DSN. You cannot use z/OS generation data groups (GDGs) to generate log or journal DSNs, but you can use the RUV variable substitution services. RUV allocates logs and journals at run time under the job user ID. You must ensure that the generated DSN can be created and accessed under your system security rules.

Number of Models to Create

You must determine the number of journal models your system requires. Separate logs from journals, and consider creating multiple models based on the anticipated volume of data. For example, consider writing larger volumes of data to tape.

Creating User Variables

The fifth implementation step is to create RUV user variables. RUV variables are a powerful mechanism for constructing journal models and recovery JCL. These variables allow you to enter variable values in a centralized location and refer to those values from a different location. RUV resolves these variables to a defined constant during execution, but their value may vary from job to job.

Types of Variables

RUV provides a predetermined set of system variables, related to job name, current date and time, and other environmental information. In addition, you can define your own variables; account numbers, DASD allocation, and device types are good variables to include.
Use of Variables

Variables have advantages over hard coded values:

- A single change to a user variable is effective for all skeleton JCL and journal models that use that variable.
- User variables can be overridden at the job level.

You can use variables in journal models and recovery JCL. Examine the journal models that you have identified, as well as the backout JCL that you want to create, and decide where variable substitution would help you.

Organizing and Coding Rule Sets

The sixth implementation step is to organize and code rule sets that describe the various decisions that you have made about recoverability and selection (Figure 10). A sample rule set is provided in member IVP#JB5 of the RUV.CNTL library (if the installation verification programs were run).

Figure 10  Rule Set Relationships
VSAM and Job Sets

You can organize the VSAM files that you have selected into VSAM sets. RUV allows you to use up to 32 characters to name a set. Good naming standards for sets help you organize and understand your data. For example, production files used in the general ledger system could be placed into a job set named PROD_GNL. The corresponding test files might be placed in a job set named TEST_GNL. See Chapter 8, “Creating Reports,” for information about how wildcard selection can assist you with reporting requirements.

Jobs and programs are also organized into job sets. Journal models, recovery job JCL, and user variables are not organized by sets. Good naming standards for these entries will help you to understand the organization of your data.

Rule Sets

After you create VSAM sets, job sets, journal models, recovery JCL, and user variables, you will combine them into one or more RUV rule sets. RUV provides the following levels of rule sets:

- system level

  The system level rule set applies to all the jobs running in a z/OS system. Only one system-level rule set is in effect at any given time; however, after a batch job is selected by a system-level rule set, the job will finish with that rule set. Activation of a new system-level rule set has no effect on currently selected jobs.

- job level

  A job-level rule set is only used by a single job and is not seen or used by any other job in the system. You may have many job-level rule sets in use simultaneously. A job-level rule set is automatically deactivated at the end of the job.

For detailed coding instructions for rule sets, see Chapter 4, “Using the Batch Journaling Facility.”

Merging and Testing Rule Sets

You will need to set up procedures for testing rule sets. The characteristics of rule sets make testing easy. System-level rule sets do not affect job steps that are already executing, and job-level rule sets affect only a single job, allowing you to isolate test jobs from the rest of the environment.
After you have activated a job-level rule set, use one of the following RUV options to test the actions:

- Use the TRY command to simulate a job.
- Run the job under a job-level rule set and ensure that the correct actions were carried out.

The results of these tests allow you to refine your recovery procedures. For example, you may want to use more DASD space for journals in a month-end job than in a daily job, or you may want to turn journaling off for certain jobs or steps that would otherwise be journaled. The results of job-level rule sets are also useful for isolating specific requirements.

Reducing the Number of Backups

A common technique in batch processing is to take backups of VSAM files that will be changed in a job and then run the update job. The backups provide an easy job restart capability, but some of them may be unnecessary after you implement RUV recoverability. You may be able eliminate some of those backups and set up procedures to recover the VSAM data sets with RUV. With RUV, you can avoid the unnecessary backups and the time required to create them.
Planning for Archive Management

The collection and management of archives is important to providing a fast, accurate recovery, especially if you are performing the recovery off-site.

Goals of Archive Management

Archive management has multiple goals:

- **reduction of number of archives**
  
  The fewer archives that you must handle, the lower the chance of error, especially when you are moving archives off-site.

- **efficient resource use**
  
  A smaller number of archives consumes less DASD or tape space. Off-site resource storage usually commands a premium price.

- **rapid recovery**
  
  Most users want to minimize the amount of time required for a recovery.

How RUV Provides CICS-Like Processing

RUV provides the same journal record capability for batch VSAM processing that CICS provides for online processing. Journal records are images of file records before or after a transaction update. These journal records include all updates, additions, and deletions of records within a data set. These changes and certain system information are also recorded in the RUV archives. RUV uses the changes reflected in the journal records to reconstruct the data set if a full recovery is needed.

RUV can potentially produce many archive files. It produces an archive file for each CICS journal extent switch, for each log stream archive, and for each step in a batch job that has recoverable VSAM files. RUV provides an accumulation process that you can use to gather and consolidate changes to selected VSAM files.

You can perform the following RUV archive management actions:

- **collect and manage RUV archives**
- **accumulate, consolidate, or separate existing archives**
- **copy and move archives to other media**
Collecting and Managing Archives

You can collect journal records and write them into RUV-controlled archives. RUV collects journal records from several sources, including a transaction server log stream or CICS journal.

The RUV accumulation function puts files, or data for a group of files, into a separate archive. If you have hundreds of files in one archive, but you need to recover one of these files quickly, the accumulation function allows you to separate that file from the others for faster recovery.

Periodic accumulation of archives helps you to optimize DASD and tape resources. This process usually involves moving recovery data from DASD to less expensive media, such as tape. You should perform the accumulation process periodically—hourly, daily, or weekly, depending on your needs. When the recovery data has been copied from many small archives to one or more larger archives, RUV provides command options to delete the small archives.

The accumulation process also allows you to group recovery data. You should place files that are normally recovered together on the same output archive. This arrangement typically speeds up a recovery because unnecessary data does not have to be passed during the recovery process. Be cautious about consolidating too many VSAM files in a single archive. A consolidated archive reduces the speed of recovery if it reduces the number of archives that can be read in parallel.

Figure 11 illustrates the value of careful planning in using the accumulation function. You specify which VSAM DSNs are to be selected and where the recovery information is to be written. RUV selects for accumulation the active archives that contain the specified VSAM files and are not accumulations.

**Figure 11  Archive to Archive Accumulation Process**

For the accumulation function, recovery data for various VSAM files is gathered from the input archives and either combined or split into multiple output archive data sets. All functions are performed with a single pass of the input data.
The accumulation process reads existing archives in parallel. You can limit the number of concurrent tape mounts if you have many tape archives to use as input to the accumulation process.

The RUV consolidated archive gathers all changes for VSAM files and writes them to an archive set, which consists of one or more output archives. You can specify the VSAM data set names to write to a particular set of archive outputs. You can choose to write the same VSAM data set to more than one archive set.

As shown in the example in Figure 12, you could have all general ledger files written to group 1, all inventory files to group 2, and all general ledger and inventory files to group 3. Then groups 1 and 2 could be retained on site for fast local recovery, while group 3 is sent to off-site storage.

**Figure 12  Sample Accumulation Strategy**

The accumulation process has many built-in safeguards. RUV retains as inactive archives those archives that are used for input to the accumulation process. RUV can purge the input archives from the RUV repository and scratch them from the z/OS catalog after the accumulation process completes; however, RUV does not purge or scratch them until all recovery information on them has been successfully written to other accumulation archives. This feature allows you to use an archive for input to several accumulation jobs. RUV allows you to create multiple copies of output archives during the accumulation process.
Accumulating Archives

The RUV Accumulation functions allow you to manage your recovery data and processes (see Figure 11 on page 66):

- You can merge archive files.
- You can separate the data for one file, or for a group of files, from all other data.
- You can separate data that is needed for forward recovery from data that is needed for backout recovery.
- You can delete input archives (from both the z/OS catalog and from the RUV repository) while you are organizing your recovery data to free resources used by the archives.

**NOTE**

Tape MOD operations are not allowed for accumulation or archive tapes.

All of these functions can be performed with a single pass of the data.

Using the accumulation process can have significant advantages for future recovery time and costs after it is optimized for your site:

- Output archive files that were created with the consolidated accumulation option retain the ability to be used for point-in-time recovery.
- Depending on the nature of your processing, accumulation can make recovery of critical data considerably faster. For example, assume that you have collected an archive with 300,000 records that cover the modifications to ten separate VSAM files. Also assume that recovery speed is more critical for one of those files (for example, the payroll file). You could use the Accumulation function to separate the archive recovery data for the payroll file from the rest by a batch job before the recovery data is actually needed for recovery. Then, when the data is needed, RUV must read and process only the payroll file data, instead of all 300,000 records from the original archive. (The data for the payroll file is probably a small subset of the original archive).

- Accumulation can reduce costs by separating data that you need to store for a longer time, or off-site, from data that you no longer need to store. For example, you may want to be able to perform a forward recovery of data from yesterday (starting from the last backup), but you probably would not need to perform a backout recovery of yesterday’s data. Accumulation would allow you to separate the forward-recovery data from the backout-recovery data, thus saving DASD or tape resources.
Accumulation can reduce costs and simplify processes by merging several archives onto a single file.

## Copying or Moving Archives

Occasionally, you may need to copy an archive from one media to another or make a copy of an archive, perhaps for off-site storage. RUV can copy the archive and can make multiple copies simultaneously. RUV registers all output archives in the repository.

During recovery processes, RUV selects only active archives for input. Consequently, you should have only one active copy of an archive; other copies should be in an inactive status.

### NOTE

Archives that have a status of PROGRES are currently open and are being updated with the Batch Journaling function. When the job step ends normally, the status is changed to ACTIVE automatically. If the batch job abnormally terminates because of a FORCE command or a power outage, the status may be left in PROGRES status. In this case, you must reset the status for the archive using the following command:

```
UPDATE ARCHIVE_FILE (name) STATUS (ACTIVE);
```

## Maintaining the Repository

Recovery data that is recorded in the repository will need to be deleted when it becomes outdated. RUV has automated this process with the PURGE command. Because RUV records recovery information that is related to dates and times, the repository must be monitored for the number of CA/CI splits and periodically reorganized.
Managing Backups

The process and frequency of backups of recoverable VSAM files will vary based on your needs. You must establish the recover time element that is required to meet any service level agreements or contracts.

Through the Backup-While-Open function, RUV supports backup of a file while the file is open to a CICS region. This function provides a major benefit if you have a 24 by 7 CICS environment.

If the VSAM data set resides on a supported intelligent storage device, RUV can interact with the BMC Software SNAPSHOT UPGRADE FEATURE (SUF) for VSAM component to create an Instant Snapshot copy of the data set. This process produces a duplicate of the data set in seconds, regardless of the size of the data set. This Instant Snapshot copy can also be restored in seconds.

When considering how often to back up a file, you should also review and understand the benefits of the RUV Batch Journaling Facility and its ability to back out VSAM updates. In many cases, RUV backout is faster and requires less processing time than running backup jobs between batch processing steps.

RUV performs prefixed and non-prefixed backups. A prefixed backup contains the sphere definition of a VSAM file. RUV uses this information during the restore of a file to rebuild the base cluster and any alternate indexes. For Backup-While-Open, RRDS, and ESDS files, RUV must use prefixed backups. RUV can use prefixed or non-prefixed backups of KSDS files.

RUV supports the use of external backups that are created by other vendor products. To automate the use of these backups, you must provide required information in a one-time definition process.

Creating an RUV backup involves determining which VSAM file or files to include for backup, building JCL with the correct space allocation, building the RUV command to specify the files to back up, and running the backup job. RUV can create multiple copies of the backup with a single pass.

If the backup files will be written to tape, you can consider stacking multiple backup files on the tape.

Backups are automatically registered in the repository, which provides RUV with the timestamp relationship that it needs to select the correct archive files during recovery.

**NOTE**

RUV can register flat-file backups of KSDSs. Registration of flat-file backups files with other VSAM file organizations is not supported.
Restoring Data Set Backups

Use RUV backups for files needing automated recovery. The backups are automatically registered and can be automatically selected when recovery is needed.

RUV can restore any backup that it has taken. You may need to restore a file to a particular historical point in time. RUV provides the information you need to determine the correct backup to use to restore the data to the point in time that you want.

If RUV has no active backup of the specified VSAM file registered in its repository, by default it attempts to restore the most recent SMS backup.

For RUV to restore another vendor’s backup, the backup must be written in key sequence in IDCAMS-standard format (flat-file format). Also, to allow RUV to automatically detect another vendor’s backup file, the file must be registered in the RUV repository. When the vendor’s backup is not registered to RUV, you must use the other vendor’s product to find the backup. You must use the other vendor’s process for restoring the file.

RUV rebuilds alternate indexes after a successful restore.

The forward recovery process automatically invokes the restore process if a VSAM data set does not exist in the z/OS catalog. In this case, RUV selects the most current backup that is allowed by the BACKUP_METHOD keyword. The default is Backup METHOD (RUV BWO SNAPSHOT SMS EXTERNAL).

If the backup was created by RUV as an Instant Snapshot copy, the restore of the data set is instantaneous.

Restoring files with RUV involves determining which VSAM files to restore, building the JCL for the restore job, building the RUV command for each file you want to restore, running the restore job, and reviewing the results of the restore job.
Obtaining RUV Reports

You can display information about the data contained in the repository by running the report for the data in which you are interested. You should use reports to view and review data that your environment has created. You should always use reporting to clarify the current state of the various types of recorded data.

Many reports accept wildcard characters as parameters. Wildcard characters allow you to restrict data to the information you need.

The content of many of the reports depends on the reporting level (summary, detail, or full) that you have specified.

For details about the REPORT command, see Chapter 8, “Creating Reports.” The REPORT command can produce the following types of RUV entries:

- “ARCHIVE_FILE Report” on page 405
- “APPLID Report” on page 407
- “BACKUP_FILE Report” on page 408
- “BACKUP_METHOD Report” on page 409
- “DEFAULT Report” on page 410
- “EXTERNAL_VENDOR Report” on page 411
- “INTERNAL_READER Report” on page 412
- “JOB_JCL Report” on page 413
- “JOB_SET Report” on page 414
- “JOURNAL_MODEL Report” on page 415
- “LOG_OF_LOGS Report” on page 416
- “LOG_STREAM Report” on page 418
- “OUTPUT_MODEL Report” on page 419
- “RULE_SET Report” on page 420
- “SYSTEM_VARIABLES Report” on page 422
- “UNIT_OF_WORK Report” on page 423
- “USER_VARIABLES Report” on page 424
- “VSAM_FILE Report” on page 425
- “VSAM_GROUP Report” on page 427
- “VSAM_GROUP_DEFINITION Report” on page 428
- “VSAM_NAME_MODEL Report” on page 429
- “VSAM_SET Report” on page 430
- “XBMID Report” on page 431

Running a report involves determining the data to display, building the JCL and command for the area to include in the report, running the job, and reviewing the content.
Using RUV for Local Recovery

If a hardware error has caused the loss of a VSAM file or made the file unreadable, you can use the forward recovery process to rebuild the lost or damaged file. You can also use forward recovery to recover a file forward to a point in time, such as the completion of a particular job. The forward recovery process uses a backup of a file and the after-image journal records to rebuild the file.

If you use RUV to perform backups, RUV can automate every element of the recovery process, including restoring the backups and determining the start and stop timestamps of the range of records to use for the recovery.

The normal construction of the start and stop timestamps is internal to RUV. A backup (as registered in the repository) contains and controls the start time needed for selecting the correct archive files. Because RUV manages all backups and archives in the repository, for a typical recovery you do not supply a start time or end time.

You can use RUV to back out (or undo) changes that have been made to a VSAM file. The backout recovery process does not use VSAM backup data sets. Instead, it uses the existing VSAM data set and applies before-image journal records to reverse the changes.

The RUV Batch Journaling Facility can perform backout recovery automatically if a program abend occurs.

**NOTE**

Automatic backout recovery is not supported from extended format ESDS data sets and IAM ESDS data sets.
Using RUV for Remote Recovery

Many data processing organizations need to be able to perform off-site recoveries. Proper planning, combined with RUV, can greatly aid the off-site recovery process. You will need to obtain a temporary password from your sales representative if you plan on testing RUV at a remote site.

At the local site, if you know that you are going off-site, use the TSO interface or batch report to obtain an inventory list of registered backups and archives for designated VSAM files. The inventory list provides you with tape volume serial numbers and allows you to identify which files are still on DASD and need to be copied to tape.

At the remote site, RUV must be installed on the new CPU. RUV is a soft-fail product. After installation, you will need to apply the temporary password using the install password process that is documented in Appendix A of the Recovery Utility for VSAM Installation and Customization Guide.

You will need to decide who can access the RUV repository and update the security access facility rules for the remote site.

After RUV has been installed, you need to bring up the RUV subsystem and reload the repository, preferably from a current backup. Reloading a backup repository will provide you with many of the elements needed for immediate recovery processing, as well as rule set and journal model definitions for later use. The RUV centralized approach to journal model definitions and user variables allows you to tailor the batch journaling environment to a new site quickly.

Any backup data sets or archive data sets that have been created since the repository backup need to be registered again. All RUV archives and prefixed backups contain enough information for repository information to be rebuilt directly from them; you do not need to register them in any special sequence. RUV depends on having cataloged data sets for any files that are allocated by data set name. Catalog the data sets that you will need for recovery as well. Use the TSO interface or a batch report to obtain an inventory list of registered backups and archives for selected VSAM files.

Using the list obtained from the TSO report at the local site, restore any non-RUV backups at the remote site. RUV-prefixed backups contain the VSAM file definitions, but if the DASD VOLSERs at the remote site are not identical to those at the local site, you must perform the IDCAMS DELETE/DEFINE process at the remote site for files that RUV has backed up. The RUV restore process reads and uses the new definitions during file restoration. This process is not required for SMS-managed VSAM files.

Rebuild any alternate indexes that are used as update paths.

Build the recovery JCL. The RUV TSO interface can be used to build recovery JCL; however, you may want to have pre-setup recovery jobs in order to be able to prioritize file recoveries.
Using RUV in CICS-Only Environments

The following sections describe the CICS features provided by RUV. See Chapter 5, “Using RUV in a CICS-Only Environment,” for more information on using RUV in CICS-only environments.

CICS Requirements

RUV uses recovery data recorded by CICS. You must specify file recovery attributes in the CICS resource definitions for files.

For files that use the Backup-While-Open feature, you must add the RUV load libraries to the CICS startup JCL in both the STEPLIB and DFHRPL concatenations. You must also define the transactions and programs to CICS. For more information, see the Recovery Utility for VSAM Installation and Customization Guide.

Journal and Log Stream Archiving

CICS records its recovery information to journal extents or z/OS log streams. This information must be captured and processed by RUV before it can be used as input to the recovery process. This capture process is referred to as archiving.

RUV reads recovery information from CICS 4.x journal extents and records the recovery information to one or more RUV archives. Information in the CICS extent does not contain an APPLID value; you must supply this data. You use the RUV ARCHIVE JOURNAL_IN command to archive a journal. CICS extents must be archived each time they switch; otherwise, the information in them may be overlaid, creating an integrity exposure.

RUV reads recovery information from Transaction Server log streams and records the recovery information to one or more RUV archives. You must archive log streams before they can be used for RUV recovery. You use the ARCHIVE command to archive a log stream. RUV also assists you with log stream management by optionally deleting data that has been captured for archive. You should be archiving log streams periodically, perhaps on a daily basis. Periodic archives reduce the amount of time required for a recovery.

All output archives are registered in the RUV repository. RUV allows you to make more than one copy of the data with a single pass of the input.
Backup-While-Open Feature

RUV automatically tracks the relationship between the CICS APPLID/FILEID and the VSAM file DSN. The relationships are established through the tie up records produced by CICS and Transaction Server. You can use the Transaction Server’s log of logs to build an inventory of VSAM files being opened and/or updated by Transaction Server regions. Transaction Server’s log of logs can be automatically scanned each time a z/OS log stream is archived if you add the log of logs name to the repository.

Backup-While-Open Feature

The Backup-While-Open feature (BWO) of RUV is especially important to CICS environments that need availability 24 hours a day, 7 days a week. RUV can back up a VSAM data set even as it is being updated by a CICS region. This process creates a fuzzy backup and requires recovery to be run against the data set when a fuzzy backup is restored. RLS files are backed up natively; the backup job can be run on any z/OS system that has access to the file in RLS mode. Backup of non-RLS files is accomplished by a combination of the RUV subsystem and RUV-supplied CICS transactions. RUV uses CICS to access the file, so CA and CI splits are not a problem. RUV supports recovery of RLS files with retained locks that are held for individual records.

You should consider making backups during periods of low activity to avoid affecting the response time of the online applications. The batch backup job must be running on the same z/OS system as the CICS region that owns the non-RLS files.

Repository Access

The RUV subsystem must be available for archive and backup registration. All output archives and backups are automatically registered in the RUV repository. The Batch Journaling Facility may be disabled if you are using RUV only for CICS file recovery.

Recovery

It is your responsibility to determine when VSAM file recovery is needed. Files being recovered must be closed to CICS during the recovery process. RUV can recover VSAM files in either a forward or backward direction. The unit-of-work (UOW) feature can support recovery of in-flight transactions.
Chapter 3 Using RUV Interfaces

This chapter provides general information about using the RUV batch interface, the RUV ISPF interface, and RUV console commands. It also provides detailed information about setting defaults for the RUV environment. The following information is included:

Using the RUV Batch Interface ............................................................. 79
  RUVZSM0 Utility JCL ................................................................. 79
  Data Set Allocation and DD Statements ....................................... 80
  Command Language Elements ............................................... 82
  Syntax Conventions ................................................................. 83
  Help for Batch Interface Problems ......................................... 84
  Commands ........................................................................... 86
  Keywords .............................................................................. 91
  Keyword Values ...................................................................... 105
  Using Masking (Wildcard) Characters ....................................... 107
  Using the COMMENT Keyword ................................................ 107
  Using the FILTER_BY Keyword .............................................. 108
  Using the HOSTCOPYMODE Keyword ....................................... 109
  Using the LOCATION Keyword .............................................. 110
  Using the MUST_COMPLETE Keyword ..................................... 110
  Using the STOP_TIME Keyword ............................................ 110
  Using the START_TIME Keyword .......................................... 111
  Using the STATUS Keyword .................................................. 112
  Using the RUV ISPF Interface ............................................... 113
  Using the RUV ISPF Interface ............................................... 113
  Starting the RUV ISPF Interface ........................................... 113
  Using the RUV Options Panel ............................................... 115
  Using Action Bar Choices ...................................................... 117
  Using Display Filters .............................................................. 119
  Using the Date Conversion Tool ........................................... 120
  Using the TOD Conversion Tool ........................................... 121
  Setting User Preferences ...................................................... 122
  Using RUV Subsystem Console Commands .......................... 123
Setting Default and Override Environment Options ............................................ 124
Establishing System and Local Job Options .................................................. 124
Using the SET DEFAULT Command ............................................................... 124
Using the SET Command ............................................................................. 125
Using the ASSOCIATED_ADDRESS_SPACE_TASKS Keyword ......................... 126
Using the BACKUP_CHANGED_ONLY Keyword .............................................. 126
Using the BACKUP_METHOD Keyword ......................................................... 126
Using the CYCLES Keyword ........................................................................ 129
Using the DATE_FORMAT Keyword ............................................................... 129
Using a User-Defined Date Format ............................................................... 130
Using the DAYS Keyword ........................................................................... 131
Using the GAP Keyword ............................................................................. 131
Using the INCREMENTAL_BACKUP_SUPPORT Keyword ............................... 132
Using the LANGUAGE Keyword ................................................................... 132
Using the LINE_COUNT Keyword ................................................................. 133
Using the MAX_CC Keyword ..................................................................... 133
Using the PAGE_WIDTH Keyword ............................................................... 133
Using the <+1> Keyword (for X37 Support) ............................................... 133
Using the READER_TASKS Keyword .............................................................. 134
Using the RECOVERY_MODE Keyword ......................................................... 134
Using the REPORT Keyword ...................................................................... 135
Using the RLS_RECOVER_PROTECTION Keyword .................................... 135
Using the SMF_TYPE Keyword ................................................................... 136
Using the SUBSYSTEM Keyword ................................................................. 136
Using the SYNTAX_CHECK Keyword ........................................................... 137
Using the TERM_CC Keyword ................................................................... 137
Using the TIME_FORMAT Keyword .............................................................. 138
Using the UPPER_CASE_TRANSLATION Keyword ...................................... 138
Using the USE_AUTOJOURNAL Keyword .................................................... 139
Using the WORK_DISK Keyword ................................................................. 141
Using the WORK_TAPE Keyword ................................................................. 141
Using the XBMID Keyword ........................................................................ 141
Viewing Default Settings ............................................................................ 142
Using RUV Default Values ......................................................................... 142
Coding SET DEFAULT Statements .............................................................. 143
Coding SET Statements ............................................................................... 144
Coding RESET DEFAULT Statements ......................................................... 145
Working with VSAM File Repository Records ............................................. 146
Using the ADD VSAM_FILE Command .......................................................... 146
Coding ADD VSAM_FILE Statements ......................................................... 146
Using the VSAM_FILE Keyword (ADD) ...................................................... 147
Using the SELECTION_EXIT Keyword (ADD) ............................................ 147
Using the APPLID_FILEID Keyword (ADD) ............................................... 147
Using Other VSAM_FILE Keywords (ADD) .............................................. 148
Using the UPDATE VSAM_FILE Command .................................................. 148
Coding UPDATE VSAM_FILE Statements .................................................... 148
Using the VSAM_FILE Keyword (UPDATE) ............................................... 149
Using the SELECTION_EXIT Keyword (UPDATE) ..................................... 149
Using the APPLID_FILEID Keyword (UPDATE) ........................................ 149
Using Other VSAM_FILE Keywords (UPDATE) ....................................... 149
Using the RUV Batch Interface

RUV provides the RUV batch utility to set up rule sets for the Batch Journaling Facility, to execute backups, restores, and recoveries, to perform repository maintenance actions, to report on information in the repository, and to perform other RUV functions. To direct utility operations, you use the RUV command language. The command language consists of control statements that contain commands, keywords, values, variables, and other elements.

RUVZSM0 Utility JCL

The RUV batch utility (program name RUVZSM0) can perform all RUV functions. To execute program RUVZSM0, customize the JCL shown in Figure 13. You identify the RUV control statements with the SYSIN DD statement. You can include the control statements in the job stream, or the SYSIN DD statement can identify an external data set that contains the control statements. Specify control statements in columns 1 through 72; column positions 73 through 80 are ignored by RUVZSM0.

You can also override the RUV subsystem name when you execute the RUV batch utility (RUVZSM0). On the EXEC statement to execute program RUVZSM0, add the PARM='/Bssid' parameter, as shown in the following example:

```//PGM1  EXEC PGM=RUVZSM0,PARM='/BRUV1'```

In this example, RUV1 is the subsystem ID.
Data Set Allocation and DD Statements

RUV uses the default DCB attributes listed in Table 1 on page 80 for the output data sets produced during RUVZSM0 utility processes.

Some commands, such as the UPDATE BACKUP_FILE and UPDATE ARCHIVE_FILE commands, do not read the data set identified on the command. They use the name you supply only to identify the record to update in the repository.

RUV automatically determines whether the keyword value is a ddname or a data set name. If the value is longer than eight characters or contains period separators, RUV knows that the value is a data set name. If the value is eight characters or less, RUV searches the task I/O table (TIOT) for a ddname that matches the value. If the TIOT search does not find a match, RUV uses the name as a data set name. The ddname (ddn8) referenced must be a valid DD statement in the JCL. Otherwise, RUV assumes the eight-character name is a data set name.

**Table 1  Default DCB Attributes for Output Data Sets**

<table>
<thead>
<tr>
<th>ddname</th>
<th>BLKSIZE</th>
<th>LRECL</th>
<th>RECFM</th>
<th>DSORG</th>
</tr>
</thead>
<tbody>
<tr>
<td>journal</td>
<td>See Note 1.</td>
<td>X</td>
<td>VBS</td>
<td>PS</td>
</tr>
<tr>
<td>(Batch Journaling Facility)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>log</td>
<td>See Note 1.</td>
<td>X</td>
<td>VBS</td>
<td>PS</td>
</tr>
<tr>
<td>(Batch Journaling Facility)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARCHIVE_OUT</td>
<td>See Note 1.</td>
<td>X</td>
<td>VBS</td>
<td>PS</td>
</tr>
<tr>
<td>BACKUP_OUT</td>
<td>See Note 1.</td>
<td>X</td>
<td>VBS</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td>See Note 3.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDS_OUT</td>
<td>80*n</td>
<td>80</td>
<td>FB</td>
<td>PO</td>
</tr>
<tr>
<td>(no PDS/E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PURGE_OUT</td>
<td>See Note 1.</td>
<td>1598</td>
<td>VB</td>
<td>PS</td>
</tr>
<tr>
<td>RPTOUT</td>
<td>121*n</td>
<td>121</td>
<td>FBA</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td>See Note 5.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUVTRACE</td>
<td>121*n</td>
<td>121</td>
<td>FBA</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td>See Note 2.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSIN</td>
<td>80*n</td>
<td>80</td>
<td>FB</td>
<td>PS</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>121*n</td>
<td>121</td>
<td>FBA</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td>See Note 2.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Note 1
This block size value depends on the device:

- For 3390 DASD, RUV uses 27998; this value allows two physical blocks per track.
- For tape, RUV uses 32760.

For block size(s) greater than these defaults, RUV writes one block per track. You should not override the default block size.

Note 2
This block size value depends on the device. You can provide a JCL block size that is a multiple of 121.

- For 3390 DASD, RUV uses 27951 (121*231); this value allows two physical blocks per track.

Note 3
RUV uses the logical record length attribute documented in the IBM® JCL Reference Manual.

For the backup data set produced by the BACKUP VSAM_IN PREFIX(NO) command, you can override the LRECL to use a size that is compatible with the VSAM record size.

Note 4
For the backup data set produced by the BACKUP VSAM_IN PREFIX(NO) command, you can override the RECFM to use a format that is compatible with the VSAM record. (When the VSAM data is Fixed, then RECFM = F or FB. When the VSAM data is Variable, then RECFM = V or VB.)

Note 5
The RUV sample selection exit uses the RPTOUT DD statement for the reports that are produced when you specify the use of a selection user exit with the SELECTION_EXIT keyword. For more information, see “Using the SELECTION_EXIT User Exit” on page 517.
The RUV command language consists of control statements that contain the following elements:

**command**
All command language statements must begin with a command that defines an action for RUV to take. For more information, see “Commands” on page 86.

**keywords**
One or more keywords may follow the command. Keywords identify the types of required and optional information that RUV needs for performing the command. For more information, see “Keywords” on page 91.

**keyword values**
One or more keyword values, enclosed by parentheses, follow a keyword. Values specify the exact information that RUV needs for performing the command. For more information, see “Keyword Values” on page 105.

**separators**
You must code a separator between multiple keyword/value combinations and between multiple keyword values. As a separator, you can use one or more blanks, a comma, a comma and one or more blanks, a line break (new line), or a comma and a line break.

**command terminator**
Use a semicolon ( ; ) to designate the end of a command.

**comment**
Use an asterisk ( * ) in column 1 to treat the entire line as a comment.
Syntax Conventions

The command charts and explanations in this user guide use the following syntax conventions:

- In the JCL syntax coding examples, required command elements are shown in bold (for illustrative purposes only).

- In the JCL usage examples, usually following the syntax examples, the required command elements are not shown in bold.

- Default values are underscored and usually appear first in the series of related values.

- Optional keywords are surrounded by braces ({}). Required keywords are not surrounded by braces. Do not code the braces in your own control statements.

- Brackets [ ] indicate a group of options; one of the options is required.

- Mutually exclusive value choices are separated by a vertical bar ( | ).

- Variables are shown in lowercase italics.

- Continuation of a series of similar variable values are shown by ellipses (...).

The following example shows the JCL syntax coding conventions used in a statement:

COMMAND KEYWORD_ONE(VALUE1 | VALUE2)
{ KEYWORD_TWO(variable1, variable2, ... ) }
{ KEYWORD_THREE(VALUE3 | VALUE4 | VALUE5) }
;

COMMAND is followed by the required KEYWORD_ONE, which has two mutually exclusive values available, with VALUE1 the default value. COMMAND also has two optional available keywords, KEYWORD_TWO and KEYWORD_THREE. KEYWORD_TWO is followed by two or more variables, variable1, variable2, and so on, which you must specify. KEYWORD_THREE has three mutually exclusive values, VALUE3, VALUE4, and VALUE5, of which VALUE4 is the default. All commands must end with a semicolon.

RUV has a default that allows you to check the syntax of your command statements. For more information, see “Using the SYNTAX_CHECK Keyword” on page 137.
Help for Batch Interface Problems

If RUV detects a problem with the syntax of any statement, it issues a series of messages (to SYSPRINT) that highlight the problem and document the correct syntax of the statement.

**Command Errors**

The first word of any RUV command must be the action being performed. If RUV cannot recognize the first word, it issues an error report (see Figure 14). The vertical bar symbol ( | ) below the command in the report typically points to the end of the invalid command. RUV then gives you a list of all supported commands.

**Figure 14  Help for Command Errors**

```
1 adi ssam_file(tstmstr):
    |
RUV202101E RUV Unsupported or invalid command.
Commands:
ACTIVATE
ADD
... [data omitted from this example]
VALIDATE
RUV202141I RUV Maximum CC set to 8 .
```

**Keyword Errors**

If RUV detects an error in a keyword, it issues an error report (see Figure 15). RUV uses the vertical bar symbol below the keyword to indicate the keyword that is in error. The symbol typically points to the end of the invalid keyword or keyword value. The next line provides a brief explanation of the error with the keyword. RUV then displays a complete list of all variations of major keywords within the command. Find the variation that best matches the task you are trying to accomplish, correct the keyword, and rerun the job.

**Figure 15  Help for Keyword Errors**

```
1    restore ssam_file(tstmstr);
    |
RUV202102E RUV Unsupported or invalid keyword.
RUV202122I RUV Command variation(s) and associated keywords for RESTORE are:
... [data omitted from this example]
VSAM_FILE, Repeatable
    44 character field
AIX, (optional)
BACKUP_IN, (optional)
NEW_NAME, (optional)
RUV202141I RUV Maximum CC set to 8 .
```
Keyword Value Errors

If RUV detects an error in a keyword value, it issues an error report (see Figure 16). RUV uses the vertical bar symbol below the keyword value to indicate the keyword that is in error. The next line provides a brief explanation of the error with the keyword value. RUV then displays a complete list of all variations of major keywords within the command. Find the variation that best matches the task you are trying to accomplish, correct the keyword value, and rerun the job.

Figure 16  Help for Keyword Value Errors

```plaintext
1   set days(x):
    |
RUV202107E RUV Indicated field must be numeric.
RUV202122I RUV Command variation(s) and associated keywords for SET DAYS are.
   DEFAULT
    ASSOCIATED_ADDRESS_SPACE_TASKS, (optional)
       3 character numeric field
   CYCLES, (optional)
       3 character numeric field
   DATE_FORMAT, (optional)
       Values are:(YYYY.JJJ | MM/DD/YYYY | DD/MM/YYYY
   DAYS, (optional)
       5 character numeric field
   INCREMENTAL_BACKUP_SUPPORT, (optional)
       Values are:(YES | NO)
   LINE_COUNT, (optional)
       8 character numeric field
   LANGUAGE, (optional)
       3 character field
```
Variable Substitution Errors

During variable substitution processing, RUV issues a series of messages (to SYSPRINT) that show the results of the variable substitution (see Figure 17). The original input string is preceded by blanks. Two equal signs (==) indicate that variable substitution was successful on this string. Two asterisks (**) indicate that variable substitution was not successful. A typical cause of the substitution failure involves a variable that was not defined and therefore did not resolve to a valid value.

Figure 17  Help for Variable Substitution Errors

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUV202420I</td>
<td>RUV BCSS Using job level rule set: V11TEST</td>
</tr>
<tr>
<td>RUV202420I</td>
<td>RUV BCSS Using system level rule set: $EXCLUDE_ALL</td>
</tr>
<tr>
<td>RUV202421I</td>
<td>RUV BCSS Selected by Job mask: ISIRAP*  Program mask: *</td>
</tr>
<tr>
<td>RUV202429E</td>
<td>RUV BCSS Error during variable substitution.</td>
</tr>
<tr>
<td>RUV202425I</td>
<td>RUV BCSS DSN=RUVRUV.QA.JNL.&lt;RUVDATE&gt;.&lt;RUVTIME&gt;,</td>
</tr>
<tr>
<td>RUV202425I</td>
<td>RUV BCSS ==DSN=RUVRUV.QA.JNL.D2009263.T1925356,</td>
</tr>
<tr>
<td>RUV202425I</td>
<td>RUV BCSS SPACE=(CYL,(&lt;PRIMARY&gt;,&lt;SECONDARY&gt;),RLSE),</td>
</tr>
<tr>
<td>RUV202425I</td>
<td>RUV BCSS ==SPACE=(CYL,(10,&lt;SECONDARY&gt;),RLSE),</td>
</tr>
<tr>
<td>RUV202425I</td>
<td>RUV BCSS DISP=(NEW,CATLG,CATLG),UNIT=SYSALLDA</td>
</tr>
<tr>
<td>RUV202425I</td>
<td>RUV BCSS ==DISP=(NEW,CATLG,CATLG),UNIT=SYSALLDA</td>
</tr>
</tbody>
</table>

Commands

RUVC commands are required as the first element of the syntax and should precede all related keywords and values or variable operands. A RUVC command is an order for an action to take place.

NOTE

To make discussions about commands more meaningful, this guide typically identifies a command by using the command verb (such as ADD or DELETE) along with the type of object on which the command operates (such as JOB_SET or VSAM_FILE).
Table 2 lists the RUV commands and their uses. For detailed information about using each command in context with a task you want to perform, see the indicated page number. For a summary of the syntax of all RUV commands, see Appendix E, “Reference Summary.”

Table 2  RUV Command Summary (Part 1 of 5)

<table>
<thead>
<tr>
<th>Command</th>
<th>Uses</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVATE RULE_SET</td>
<td>establishes a job-level environment and the associated rules by which VSAM files can be selected for batch journaling</td>
<td>198</td>
</tr>
<tr>
<td>ADD EXTERNAL_VENDOR</td>
<td>creates a descriptive name associated with a vendor or process performed by the vendor. A vendor will be supplied data. This data is intended to be used in a cooperative effort to satisfy a user’s needs</td>
<td>187</td>
</tr>
<tr>
<td>ADD INTERNAL_READER</td>
<td>creates a rule set element that controls the processing of recovery JCL generated when a job abends</td>
<td>184</td>
</tr>
<tr>
<td>ADD JOB_JCL</td>
<td>creates a rule set element that contains job JCL to be executed</td>
<td>183 330</td>
</tr>
<tr>
<td>ADD JOB_SET</td>
<td>creates a rule set element that controls logging, journaling, and error recovery in relation to job or program names</td>
<td>161</td>
</tr>
<tr>
<td>ADD JOURNAL_MODEL</td>
<td>creates a rule set element that controls the creation of journals and logs during job step execution</td>
<td>166</td>
</tr>
<tr>
<td>ADD LOG_OF_LOGS</td>
<td>adds a log of logs record to the repository</td>
<td>240</td>
</tr>
<tr>
<td>ADD LOG_STREAM</td>
<td>adds a log stream record to the repository</td>
<td>236</td>
</tr>
<tr>
<td>ADD OUTPUT_MODEL</td>
<td>adds a record to the repository for selected device storage</td>
<td>615</td>
</tr>
<tr>
<td>ADD RULE_SET</td>
<td>creates and records in the repository a logical combination of user variables, and job and VSAM sets, along with any referenced models that, when activated, are read into memory and formed into a set of rules</td>
<td>193</td>
</tr>
<tr>
<td>ADD SEQ_FILE</td>
<td>adds a sequential DSN record and its associated information to the repository</td>
<td>688</td>
</tr>
<tr>
<td>ADD SEQ_GROUP_DEFINITION</td>
<td>adds a sequential group definition to the repository</td>
<td>688</td>
</tr>
<tr>
<td>ADD USER_VARIABLES</td>
<td>creates a set of user variables to be used as part of a rule set and during the substitution phase of abend job submission</td>
<td>171</td>
</tr>
<tr>
<td>ADD VSAM_FILE</td>
<td>adds a VSAM DSN record and its associated information to the repository</td>
<td>147</td>
</tr>
<tr>
<td>ADD VSAM_GROUP_DEFINITION</td>
<td>adds a VSAM group definition to the repository</td>
<td>601</td>
</tr>
<tr>
<td>ADD VSAM_NAME_MODEL</td>
<td>adds a VSAM name model record (for use in creating Instant Snapshot copies) to the repository</td>
<td>305</td>
</tr>
<tr>
<td>ADD VSAM_SET</td>
<td>adds a set of VSAM journaling rules that determine which journal records, if any, to write to the journal on an individual DSN basis</td>
<td>159</td>
</tr>
<tr>
<td>ARCHIVE ACCUM</td>
<td>identifies a set of input recovery archive data (for specified VSAM files) to accumulate and write to one or more output archives</td>
<td>250</td>
</tr>
</tbody>
</table>
Table 2  RUV Command Summary (Part 2 of 5)

<table>
<thead>
<tr>
<th>Command</th>
<th>Uses</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIVE ARCHIVE_IN</td>
<td>creates another copy of an archive file or transfers an archive to different media</td>
<td>260</td>
</tr>
<tr>
<td>ARCHIVE JOURNAL_IN</td>
<td>reads CICS journals and processes VSAM recovery data to RUV archives</td>
<td>227</td>
</tr>
<tr>
<td>ARCHIVE LOG_STREAM_IN</td>
<td>reads CICS logs and journals from a log stream</td>
<td>233</td>
</tr>
<tr>
<td>ARCHIVE RPCV_IN</td>
<td>reads RPCV batch journaling recovery data and processes it to RUV archives</td>
<td>272</td>
</tr>
<tr>
<td>BACKUP REPOSITORY</td>
<td>writes a copy of all RUV records in the repository to one or more sequential data sets</td>
<td>573</td>
</tr>
<tr>
<td>BACKUP SEQ_GROUP</td>
<td>copies a group of sequential data sets to one or more sequential data sets</td>
<td>692</td>
</tr>
<tr>
<td>BACKUP SEQ_IN</td>
<td>copies a single sequential data set to one or more sequential data sets</td>
<td>693</td>
</tr>
<tr>
<td>BACKUP VSAM_GROUP</td>
<td>copies a group of VSAM base cluster to one or more sequential data sets</td>
<td>295</td>
</tr>
<tr>
<td>BACKUP VSAM_IN</td>
<td>copies one VSAM base cluster to one or more sequential data sets</td>
<td>292</td>
</tr>
<tr>
<td>DELETE ARCHIVE_FILE</td>
<td>removes one or more archive file records from the repository</td>
<td>268</td>
</tr>
<tr>
<td>DELETE BACKUP_FILE</td>
<td>removes one or more backup file records from the repository</td>
<td>367</td>
</tr>
<tr>
<td>DELETE EXTERNAL_VENDOR</td>
<td>removes a descriptive name associated with a vendor or process performed by a vendor.</td>
<td>184</td>
</tr>
<tr>
<td>DELETE INTERNAL_READER</td>
<td>removes internal reader rule set element records from the repository</td>
<td>212</td>
</tr>
<tr>
<td>DELETE JOB_JCL</td>
<td>removes job JCL rule set element records from the repository</td>
<td>212</td>
</tr>
<tr>
<td>DELETE JOB_SET</td>
<td>removes job set rule set element records from the repository</td>
<td>212</td>
</tr>
<tr>
<td>DELETE JOURNAL_MODEL</td>
<td>removes journal model rule set element records from the repository.</td>
<td>212</td>
</tr>
<tr>
<td>DELETE LOG_OF_LOGS</td>
<td>removes log of logs records from the repository</td>
<td>240</td>
</tr>
<tr>
<td>DELETE LOG_STREAM</td>
<td>removes log stream records from the repository</td>
<td>237</td>
</tr>
<tr>
<td>DELETE OUTPUT_MODEL</td>
<td>removes an output_model definition from the repository.</td>
<td>617</td>
</tr>
<tr>
<td>DELETE RULE_SET</td>
<td>removes a rule set definition record from the repository</td>
<td>212</td>
</tr>
<tr>
<td>DELETE SEQ_FILE</td>
<td>removes sequential DSN records from the repository</td>
<td>695</td>
</tr>
<tr>
<td>DELETE SEQ_GROUP</td>
<td>removes sequential files within this group from the repository</td>
<td>695</td>
</tr>
<tr>
<td>DELETE SEQ_GROUP_DEFINITION</td>
<td>removes a sequential group definition from the repository</td>
<td>695</td>
</tr>
<tr>
<td>DELETE USER_VARIABLES</td>
<td>removes a set of user variables from the repository</td>
<td>212</td>
</tr>
<tr>
<td>DELETE VSAM_FILE</td>
<td>removes VSAM DSN records from the repository</td>
<td>150</td>
</tr>
<tr>
<td>DELETE VSAM_GROUP</td>
<td>removes VSAM files within this group from the repository</td>
<td>619</td>
</tr>
<tr>
<td>DELETE VSAM_GROUP_DEFINITION</td>
<td>removes a VSAM group definition from the repository</td>
<td>618</td>
</tr>
<tr>
<td>Command</td>
<td>Uses</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>DELETE VSAM_NAME_MODEL</td>
<td>removes a VSAM name model record (for use in creating Instant Snapshot copies) from the repository</td>
<td>305</td>
</tr>
<tr>
<td>DELETE VSAM_SET</td>
<td>removes a set of VSAM rules from the repository</td>
<td>212</td>
</tr>
<tr>
<td>ENTER_IDCAMS</td>
<td>initiates the passing of control statements to IDCAMS</td>
<td>514</td>
</tr>
<tr>
<td>EXIT_IDCAMS</td>
<td>terminates the passing of control statements to IDCAMS</td>
<td>514</td>
</tr>
<tr>
<td>PRINT ARCHIVE_IN</td>
<td>prints records from an archive</td>
<td>274</td>
</tr>
<tr>
<td>PRINT LOG_STREAM_IN</td>
<td>prints records from a log stream</td>
<td>234</td>
</tr>
<tr>
<td>PRINT REPOSITORY</td>
<td>prints records from a repository</td>
<td>577</td>
</tr>
<tr>
<td>PURGE REPOSITORY</td>
<td>purges records from the repository</td>
<td>588</td>
</tr>
<tr>
<td>RECOVER BACKOUT</td>
<td>uses before-image journal records to back out changes to one or more identified VSAM files, returning the files to a previous state</td>
<td>489</td>
</tr>
<tr>
<td>RECOVER COMPLETED_UNIT_OF_WORK</td>
<td>uses after-image journal records and a previous backup, if needed or requested, to rebuild a VSAM file by applying completed transactions and recording information about in-flight transactions</td>
<td>483</td>
</tr>
<tr>
<td>RECOVER FORWARD</td>
<td>uses after-image journal records and a previous backup, if needed or requested, to rebuild a VSAM file</td>
<td>473</td>
</tr>
<tr>
<td>RECOVER REPOSITORY FORWARD</td>
<td>recovers repository using RUV archives, SMF records, and repository backup</td>
<td>568</td>
</tr>
<tr>
<td>REGISTER ARCHIVE_IN</td>
<td>reads an existing RUV archive and rebuilds repository information about it</td>
<td>262</td>
</tr>
<tr>
<td>REGISTER BACKUP_FILE</td>
<td>builds information relating to non-RUV and non-SMS backups in the repository</td>
<td>356</td>
</tr>
<tr>
<td>REGISTER BACKUP_IN</td>
<td>reads an existing RUV backup and rebuilds repository information about it</td>
<td>362</td>
</tr>
<tr>
<td>REGISTER RESTORE_JOB_JCL</td>
<td>registers an external backup with information about the job JCL record that RUV should use to restore the backup automatically</td>
<td>357</td>
</tr>
<tr>
<td>REGISTER RESTORE_METHOD</td>
<td>causes RUV to register a trigger point for an external backup so that if the backup needs to be restored, RUV issues a WTOR to request that the operator handle the restore</td>
<td>359</td>
</tr>
<tr>
<td>REGISTER LOG_OF_LOGS</td>
<td>reads a z/OS log of logs log stream and records the DSN and APPLID_FILEID information into the repository</td>
<td>239</td>
</tr>
<tr>
<td>REORGANIZE VSAM_GROUP</td>
<td>reorganizes defined VSAM groups</td>
<td>444</td>
</tr>
<tr>
<td>REORGANIZE VSAM_IN</td>
<td>reorganizes individual VSAM files</td>
<td>440</td>
</tr>
<tr>
<td>REPORT</td>
<td>creates a report of information from the repository filtered by specified parameters (fields)</td>
<td>401</td>
</tr>
<tr>
<td>REPORT AUDIT</td>
<td>produces reports of changes to VSAM data sets by processing Archives alone</td>
<td>432</td>
</tr>
<tr>
<td>RESET DEFAULT</td>
<td>deletes the default record written to the repository by the SET DEFAULT command</td>
<td>142</td>
</tr>
</tbody>
</table>
### Table 2  RUV Command Summary (Part 4 of 5)

<table>
<thead>
<tr>
<th>Command</th>
<th>Uses</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTORE REPOSITORY</td>
<td>copies an RUV backup of the repository back into the RUV repository files</td>
<td>574</td>
</tr>
<tr>
<td>RESTORE SEQ_FILE</td>
<td>copies backup RUV files to sequential clusters</td>
<td>703</td>
</tr>
<tr>
<td>RESTORE SEQ_GROUP</td>
<td>copies backup RUV group files to sequential clusters</td>
<td>704</td>
</tr>
<tr>
<td>RESTORE VSAM_FILE</td>
<td>copies backup RUV files to VSAM clusters</td>
<td>457</td>
</tr>
<tr>
<td>RESTORE VSAM_GROUP</td>
<td>copies backup RUV group files to VSAM clusters</td>
<td>605</td>
</tr>
<tr>
<td>SET</td>
<td>temporarily controls various options during a job step</td>
<td>125</td>
</tr>
<tr>
<td>SET DEFAULT</td>
<td>permanently changes various default options by writing a default record in the repository</td>
<td>124</td>
</tr>
<tr>
<td>STORE INTO_FILE</td>
<td>store information into the specified file for later retrieval</td>
<td>332</td>
</tr>
<tr>
<td>SUBMIT BACKUP</td>
<td>submit the specified backup job JCL record for execution</td>
<td>341</td>
</tr>
<tr>
<td>TEST DELAY</td>
<td>allows you to introduce a delay between the time when the process is submitted for execution and the time when the process starts executing</td>
<td>515</td>
</tr>
<tr>
<td>TRY</td>
<td>allows you to see how a job, program, or data set name combination will be interpreted by the active rule set</td>
<td>201</td>
</tr>
<tr>
<td>UPDATE ARCHIVE_FILE</td>
<td>updates fields in an archive registration record in the repository, allowing you to change the status, location, or comment fields</td>
<td>264</td>
</tr>
<tr>
<td>UPDATE BACKUP_FILE</td>
<td>updates fields in a backup registration record in the repository</td>
<td>364</td>
</tr>
<tr>
<td>UPDATE EXTERNAL_VENDOR</td>
<td>updates a name associated with a vendor or process performed by the vendor.</td>
<td>184</td>
</tr>
<tr>
<td>UPDATE INTERNAL_READER</td>
<td>updates a rule set element that controls the processing of recovery JCL generated when a job abends</td>
<td>185</td>
</tr>
<tr>
<td>UPDATE JOB_JCL</td>
<td>updates a rule set element that contains job JCL to be executed</td>
<td>183</td>
</tr>
<tr>
<td></td>
<td></td>
<td>330</td>
</tr>
<tr>
<td>UPDATE JOURNAL_MODEL</td>
<td>updates a rule set element that controls the creation of journals and logs during job step execution</td>
<td>166</td>
</tr>
<tr>
<td>UPDATE LOG_OF_LOGS</td>
<td>updates fields in a log of logs record in the repository</td>
<td>240</td>
</tr>
<tr>
<td>UPDATE LOG_STREAM</td>
<td>updates fields in a log stream record in the repository</td>
<td>238</td>
</tr>
<tr>
<td>UPDATE OUTPUT_MODEL</td>
<td>updates a record in the repository for selected device storage</td>
<td>617</td>
</tr>
<tr>
<td>UPDATE SEQ_FILE</td>
<td>updates fields in a DSN record in the repository</td>
<td>710</td>
</tr>
<tr>
<td>UPDATE SEQ_GROUP</td>
<td>updates a sequential group in the repository</td>
<td>710</td>
</tr>
<tr>
<td>UPDATE SEQ_GROUP_DEFINITION</td>
<td>updates a sequential group definition in the repository</td>
<td>710</td>
</tr>
<tr>
<td>UPDATE VSAM_FILE</td>
<td>updates fields in a DSN record in the repository</td>
<td>148</td>
</tr>
<tr>
<td>UPDATE VSAM_GROUP</td>
<td>updates a VSAM group in the repository</td>
<td>619</td>
</tr>
<tr>
<td>UPDATE VSAM_GROUP_DEFINITION</td>
<td>updates a VSAM group definition in the repository</td>
<td>618</td>
</tr>
<tr>
<td>UPDATE VSAM_NAME_MODEL</td>
<td>updates a VSAM name model record (for use in creating Instant Snapshot copies) in the repository</td>
<td>305</td>
</tr>
</tbody>
</table>
Keywords

Keywords identify the types of information that RUV needs for performing a command. Some keywords are required with the command and must be coded in the entire command statement. Some keywords are required but have default values. If you omit the keyword with a default from the command statement, RUV will use the default value automatically. Some keywords are optional; you can omit or include them to meet your needs. You can code keywords in any order.

There are some useful keyword conventions to help you understand how various commands work. These conventions apply to archives, backups, journals, log streams, RPCV journals, PDSs, and output from the PURGE command.

If a keyword ends with ‘_IN’, then the command will need the actual resource to process correctly. For example, ARCHIVE_IN(PROD.GNL.ARCH.V0001051) will need to allocate and read the data set named PROD.GNL.ARCH.V0001051.

If a keyword ends with ‘_OUT’, then the resource will be used for output. A keyword ending with ‘_FILE’ identifies a resource name but does not necessarily create or access a physical file.

Table 3 lists RUV keywords, a brief description of the keyword, the commands with which you can use the keyword, and one or more pages where you can find information about using the keyword.

For information about values, see “Keyword Values” on page 105.

Table 2  RUV Command Summary (Part 5 of 5)

<table>
<thead>
<tr>
<th>Command</th>
<th>Uses</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALIDATE REPOSITORY</td>
<td>detect and correct errors in the repository by deleting invalid records</td>
<td>588</td>
</tr>
<tr>
<td>VALIDATE SUBSYSTEM</td>
<td>validates a subsystem relating to control block linkage and validity</td>
<td>554</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
<td>Related Commands</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>&lt;+1&gt;</td>
<td>sets the value of the &lt;+1&gt; variable after restoring the RUV repository from a backup; RUV increments this value by 1 before substituting the value in a DSN</td>
<td>SET</td>
</tr>
<tr>
<td>ACCUM</td>
<td>identifies that user-selected set of input recovery data for specified VSAM files will be accumulated and written to a set of associated output archives</td>
<td>ARCHIVE ACCUM</td>
</tr>
<tr>
<td>AIX</td>
<td>specifies whether to rebuild alternate indexes during a restore</td>
<td>RESTORE VSAM_FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER FORWARD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REORGANIZE VSAM_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REORGANIZE VSAM_GROUP</td>
</tr>
<tr>
<td>APPLID</td>
<td>identifies a relationship to the VSAM data set name</td>
<td>ARCHIVE JOURNAL_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT APPLID</td>
</tr>
<tr>
<td>APPLID_FILEID</td>
<td>identifies a relationship to a VSAM file by providing an APPLID and a FILEID</td>
<td>ADD VSAM_FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE VSAM_FILE</td>
</tr>
<tr>
<td>ARCHIVE_FILE</td>
<td>identifies the data set that contains archive data</td>
<td>UPDATE ARCHIVE_FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE ARCHIVE_FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT ARCHIVE_FILE</td>
</tr>
<tr>
<td>ARCHIVE_IN</td>
<td>identifies any file containing journal data created by RUV</td>
<td>ARCHIVE ACCUM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ARCHIVE ARCHIVE_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REGISTER ARCHIVE_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRINT ARCHIVE_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER BACKOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER FORWARD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT AUDIT</td>
</tr>
<tr>
<td>ARCHIVE_OUT</td>
<td>identifies a file to contain output journal data</td>
<td>ARCHIVE JOURNAL_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ARCHIVE LOG_STREAM_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ARCHIVE ACCUM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ARCHIVE ARCHIVE_IN</td>
</tr>
<tr>
<td>ASSOCIATED_ADDRESS_</td>
<td>controls the number of associated address spaces that can be attached</td>
<td>SET DEFAULT</td>
</tr>
<tr>
<td>SPACE_TASKS</td>
<td></td>
<td>SET</td>
</tr>
<tr>
<td>BACKUP_CHANGED_ONLY</td>
<td>controls whether RUV backs up only the files that have changed in a group</td>
<td>SET DEFAULT</td>
</tr>
<tr>
<td>BACKUP_FILE</td>
<td>identifies a backup data set that was not created by RUV or SMS.</td>
<td>UPDATE BACKUP_FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REGISTER BACKUP_FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE BACKUP_FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT BACKUP_FILE</td>
</tr>
</tbody>
</table>
### Table 3  RUV Keywords (Part 2 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP_IN</td>
<td>identifies a data set that was created during an RUV BACKUP command</td>
<td>REGISTER BACKUP_IN, RESTORE SEQ_FILE, RESTORE VSAM_FILE, RECOVER FORWARD, RESTORE REPOSITORY, CURRENT_BACKUP</td>
<td>362</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>461</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>493</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>574</td>
</tr>
<tr>
<td>BACKUP_METHOD</td>
<td>controls how backup methods are used during backup and restore processes</td>
<td>SET DEFAULT, SET, REPORT</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>126</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>371</td>
</tr>
<tr>
<td>BACKUP_OUT</td>
<td>identifies the data set to contain output backup data</td>
<td>BACKUP SEQ_IN, BACKUP VSAM_IN, BACKUP REPOSITORY, REORGANIZE VSAM_IN</td>
<td>293</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>576</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>441</td>
</tr>
<tr>
<td>BACKUP_OUT_TYPE</td>
<td>identifies the designated device for storage</td>
<td>ADD OUTPUT_MODEL, UPDATE OUTPUT_MODEL</td>
<td>615</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>617</td>
</tr>
<tr>
<td>BACKUP_REGISTRATION</td>
<td>allows the user to integrate vendor software at the point RUV journal</td>
<td>ADD RULE_SET, ADD EXTERNAL_VENDOR, UPDATE EXTERNAL_VENDOR, DELETE EXTERNAL_VENDOR</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>registration is being done. An exit program must be written to receive the</td>
<td></td>
<td>188</td>
</tr>
<tr>
<td></td>
<td>data provided by RUV at the point(s) RUV processing invokes the program. The</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>default is NO.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUFND</td>
<td>specifies the number of data buffers to use</td>
<td>RECOVER FORWARD, RECOVER BACKOUT</td>
<td>478</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>491</td>
</tr>
<tr>
<td>BUFNI</td>
<td>specifies the number of index buffers to use</td>
<td>RECOVER FORWARD, RECOVER BACKOUT</td>
<td>478</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>491</td>
</tr>
<tr>
<td>CLASS</td>
<td>provides the same information as a z/OS JCL CLASS variable</td>
<td>ADD INTERNAL_READER, UPDATE INTERNAL_READER</td>
<td>185</td>
</tr>
<tr>
<td>COMMENT</td>
<td>provides a 200-byte (four 50-byte lines each) field for user documentation</td>
<td>all ADD commands, all ARCHIVE commands, BACKUP SEQ_GROUP, BACKUP SEQ_IN, BACKUP VSAM_GROUP, BACKUP VSAM_IN</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>purposes</td>
<td>all REGISTER commands except LOG_OF_LOGS, all UPDATE commands, all REORGANIZE commands</td>
<td>629</td>
</tr>
</tbody>
</table>
### RUV Keywords (Part 3 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
</table>
| COMPLETE_COPY      | copies all original data from the log stream in addition to copying the recovery data into an RUV format | ARCHIVE JOURNAL_IN  
ARCHIVE RPCV_IN | 228  
272 |
| CONSOLIDATED       | specifies that you want to write a data set to a consolidated archive     | ARCHIVE ACCUM | 254 |
| COUNT              | selects a range of records by specifying the range from a relative record number to a relative record number | PRINT ARCHIVE_IN  
PRINT REPOSITORY | 274  
577 |
| CURRENT_BACKUP     | automatically selects the most current backup as part of restore process and recover forward process. | RECOVER_FORWARD  
RESTORE SEQ_FILE  
RESTORE VSAM_FILE | 473  
457 |
| CYCLES             | specifies groupings of related data files (cycles) that contain a complete recoverable VSAM file by providing the number of cycles you want to retain | SET DEFAULT  
SET  
UPDATE SEQ_FILE  
UPDATE VSAM_FILE | 124  
125  
149 |
| DATE_FORMAT        | controls the layout structure of displayed dates                           | SET DEFAULT  
SET | 124  
125 |
| DAYS               | specifies a number, nnn, that directs processing to retain repository information for nnn days | SET DEFAULT  
SET  
REGISTER LOG_OF_LOGS  
ADD SEQ_FILE  
UPDATE SEQ_FILE  
UPDATE VSAM_FILE | 124  
125  
242  
149 |
| DDNAME             | specifies the DD name of a statement in the JCL that identifies a data set to be used by RUV | ADD JOURNAL_MODEL  
UPDATE JOURNAL_MODEL  
ADD INTERNAL_READER  
UPDATE INTERNAL_READER | 169  
185 |
| DELAY              | specifies the length of time, in seconds, for RUV to pause before continuing with execution | TEST DELAY | 515 |
| DEFAULT            | refers to the values that are used when no value is provided for a specific keyword | SET DEFAULT  
REPORT DEFAULT | 124  
410 |
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>allows removal of information referred to in the command being requested</td>
<td>ARCHIVE LOG_STREAM_IN PRINT LOG_STREAM_IN REGISTER LOG_OF_LOGS</td>
<td>231</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>234</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>241</td>
</tr>
<tr>
<td>DSN</td>
<td>identifies the particular data set name</td>
<td>TRY</td>
<td>201</td>
</tr>
<tr>
<td>END_DATA</td>
<td>indicates the end of the input definitions</td>
<td>ADD USER_VARIABLES ADD JOB_JCL</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>183</td>
</tr>
<tr>
<td>ESOTERIC</td>
<td>specifies the name of an esoteric device to use for Instant Snapshot copy processing</td>
<td>SET SET DEFAULT</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>124</td>
</tr>
<tr>
<td>EXTERNAL_VENDOR</td>
<td>indicates a user-written exit program that identifies an external vendor</td>
<td>ADD EXTERNAL_VENDOR ADD RULE_SET DELETE EXTERNAL_VENDOR UPDATE EXTERNAL_VENDOR REPORT</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>183</td>
</tr>
<tr>
<td>FILTER_BY</td>
<td>filters the selection of records by location, status, start time, and stop time</td>
<td>DELETE BACKUP_FILE DELETE ARCHIVE_FILE REPORT BACKUP_FILE REPORT ARCHIVE_FILE UPDATE BACKUP_FILE UPDATE ARCHIVE_FILE</td>
<td>108</td>
</tr>
<tr>
<td>GAP</td>
<td>controls actions when RUV detects missing data during logstream processing</td>
<td>SET DEFAULT SET ARCHIVE LOG_STREAM_IN</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>124</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>230</td>
</tr>
<tr>
<td>HOSTCOPYMODE</td>
<td>specifies for Instant Snapshot copies whether the task will share control or obtain exclusive control of the data set that is being copied</td>
<td>ADD VSAM_NAME_MODEL UPDATE VSAM_NAME_MODEL</td>
<td>109</td>
</tr>
<tr>
<td>_INCREMENTAL_BACKUP_SUPPORT</td>
<td>controls the reset of the data-set-changed indicator for a VSAM file</td>
<td>SET DEFAULT SET</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>125</td>
</tr>
<tr>
<td>INTERNAL_READER</td>
<td>provides a descriptive name for the internal reader</td>
<td>ADD INTERNAL_READER ADD JOB_SET REPORT INTERNAL_READER SUBMIT BACKUP</td>
<td>164</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>184</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>412</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>341</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
<td>Related Commands</td>
<td>Pages</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>INVENTORY_ONLY</td>
<td>allows you to check for desired results before actually performing a process</td>
<td>ARCHIVE, BACKUP SEQ_GROUP, BACKUP SEQ_IN, BACKUP VSAM_GROUP, RECOVER BACKOUT, RECOVER_FORWARD, RESTORE SEQ_FILE, RESTORE SEQ_GROUP, RESTORE VSAM_GROUP, UPDATE SEQ_GROUP, UPDATE VSAM_GROUP, SUBMIT BACKUP, REORGANIZE VSAM_IN, REORGANIZE VSAM_GROUP, REPORT AUDIT</td>
<td>251</td>
</tr>
<tr>
<td>INTO_FILE</td>
<td>specifies the data set name or partitioned data set member to contain stored information</td>
<td>STORE</td>
<td>332</td>
</tr>
<tr>
<td>JOB</td>
<td>identifies the particular job</td>
<td>TRY</td>
<td>201</td>
</tr>
<tr>
<td>JOB_ABEND</td>
<td>specifies whether to submit recovery JCL if a job logging under control of RUV abends</td>
<td>ADD JOB_SET</td>
<td>163</td>
</tr>
<tr>
<td>JOB_JCL</td>
<td>specifies the name of the recovery JCL to be submitted or stored in a PDS for later execution</td>
<td>ADD JOB_SET, ADD JOB_JCL, DELETE JOB_JCL, REPORT JOB_JCL, SUBMIT BACKUP</td>
<td>163, 182, 212, 413, 341</td>
</tr>
<tr>
<td>JOB_RULE</td>
<td>specifies a job name and program name that will trigger batch journaling</td>
<td>ADD JOB_SET</td>
<td>161</td>
</tr>
<tr>
<td>JOB_SET</td>
<td>creates a rule set element that controls logging, journal, and error recovery in relation to job or program names</td>
<td>ADD JOB_SET, ADD RULE_SET, DELETE JOB_SET, REPORT JOB_SET</td>
<td>161, 192, 212, 414</td>
</tr>
<tr>
<td>JOURNAL</td>
<td>specifies the journaling action to take</td>
<td>ADD JOB_SET</td>
<td>162</td>
</tr>
<tr>
<td>JOURNAL_ABEND</td>
<td>specifies the action to take in the event of a journaling failure</td>
<td>ADD JOB_SET</td>
<td>163</td>
</tr>
<tr>
<td>JOURNAL_IN</td>
<td>identifies the input source of the CICS journal data</td>
<td>ARCHIVE JOURNAL_IN</td>
<td>228</td>
</tr>
</tbody>
</table>
### Table 3  RUV Keywords (Part 6 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
</table>
| JOURNAL_MODEL          | specifies the name of a journal model that contains the related attributes used to allocate QSAM journal and log files | ADD JOURNAL_MODEL<br>DELETE JOURNAL_MODEL<br>REPORT JOURNAL_MODEL<br>UPDATE JOURNAL_MODEL | 166  
|                        |                                                                             |                                                                                  | 212  
|                        |                                                                             |                                                                                  | 415  |
| JOURNAL_REGISTRATION   | allows the user to integrate vendor software at the point RUV journal registration is being done. The exit program must be written to receive the data provided by RUV at the point(s) RUV processing invokes the program. The default is NO. | ADD RULE_SET<br>ADD EXTERNAL_VENDOR<br>UPDATE EXTERNAL_VENDOR<br>DELETE EXTERNAL_VENDOR | 192  
|                        |                                                                             |                                                                                  | 189  |
| LANGUAGE               | allows for future national language support                                 | SET DEFAULT<br>SET                                                               | 124  
|                        | Currently, English is supported.                                             |                                                                                  | 125  |
| LAST_IMAGE             | selects the last journal image for archiving                                | ARCHIVE ACCUM                                                                    | 255  |
| LINE_COUNT             | controls the number (nnn) of lines per page                                  | SET DEFAULT<br>SET                                                               | 124  
|                        | nnn must be in the range from 15 to 9999999; 56 is the default.              |                                                                                  | 125  |
| LOCATION               | provides an 8-character documentation field for your own use; can be used as a filter for ISPF and batch reports | all ARCHIVE commands<br>all BACKUP commands<br>DELETE ARCHIVE_FILE<br>all REGISTER commands except LOG_OF_LOGS<br>REPORT SEQ_FILE<br>REPORT VSAM_FILE<br>UPDATE ARCHIVE_FILE<br>UPDATE BACKUP_FILE<br>all REORGANIZE commands | 110  |
| LOG                    | specifies the logging action to take                                         | ADD JOB_SET                                                                      | 162  |
## Table 3  RUV Keywords (Part 7 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_OF_LOGS</td>
<td>identifies a coupling facility structure that CICS Transaction Server uses to record information related to VSAM open and close request; RUV uses this structure to identify all VSAM file data set names and their related APPLID and FILEID names</td>
<td>ADD LOG_OF_LOGS DELETE LOG_OF_LOGS REGISTER LOG_OF_LOGS UPDATE LOG_OF_LOGS REPORT LOG_OF_LOGS</td>
<td>241</td>
</tr>
<tr>
<td>LOG_STREAM</td>
<td>identifies the source that contains the journal data before-images and after-images</td>
<td>ADD LOG_STREAM DELETE LOG_STREAM REPORT LOG_STREAM UPDATE LOG_STREAM</td>
<td>237</td>
</tr>
<tr>
<td>LOG_STREAM_IN</td>
<td>identifies the input source that contains the journal data before-images and after-images</td>
<td>ARCHIVE LOG_STREAM_IN PRINT LOG_STREAM_IN</td>
<td>230-234</td>
</tr>
<tr>
<td>LSR_POOL</td>
<td>specifies whether or not to use Local Shared Resource (LSR) Pools to improve performance</td>
<td>RECOVER FORWARD RECOVER BACKOUT</td>
<td>479-492</td>
</tr>
<tr>
<td>MEMBER</td>
<td>specifies the name of a PDS member that RUV uses to store the JCL for a backout recovery job</td>
<td>ADD INTERNAL_READER UPDATE INTERNAL_READER</td>
<td>185</td>
</tr>
<tr>
<td>MERGE</td>
<td>controls whether existing records are replaced during a repository restore process</td>
<td>RESTORE REPOSITORY</td>
<td>575</td>
</tr>
<tr>
<td>MUST_COMPLETE</td>
<td>controls whether a process for a specified command or keyword must complete successfully</td>
<td>all ARCHIVE commands</td>
<td>110</td>
</tr>
<tr>
<td>NEW_NAME</td>
<td>allows alternative data sets to be used in the recover forward process or the restore process</td>
<td>RECOVER FORWARD RESTORE SEQ_FILE RESTORE VSAM_FILE</td>
<td>481-459</td>
</tr>
<tr>
<td>OUTPUT_MODEL</td>
<td>identifies the output model that contains the allocation information that you want to use for the output backup data set</td>
<td>REORGANIZE VSAM_GROUP</td>
<td>445</td>
</tr>
</tbody>
</table>
## Table 3  RUV Keywords (Part 8 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
</table>
| PAGE_WIDTH               | specifies a report page width number, nnn, from 30 to 120 characters per line; the default is 120 | SET DEFAULT  
SET                                                                                   | 124  
125 |
| PARAMETER                | specifies a user field (1—64 bytes) for subsequent access or display        | REGISTER RESTORE_JOB_JCL  
REGISTER RESTORE_METHOD                                                                 | 357  
359 |
| PASSWORD_EXPIRATION_CC   | specifies the condition code to issue before the product password expires   | SET  
SET DEFAULT                                                                             | 125  
124 |
| PDS_OUT                  | provides the name of a PDS to receive JCL generated by RUV processing       | ADD INTERNAL_READER  
UPDATE INTERNAL_READER                                                                 | 185 |
| PROTECTION_LEVEL         | provides RUV BSAM or QSAM use for batch journaling; balances the level of safety against the level of performance | ADD JOURNAL_MODEL  
UPDATE JOURNAL_MODEL                                                                    | 168 |
| PREFIX                   | indicates the existence of a prefix on each record of the backup copy of the VSAM file | BACKUP VSAM_IN                                                                 | 292 |
| PROGRAM                  | identifies the particular program to test                                   | TRY                                                                                       | 201 |
| PTF                      | provides information from the RUV load library and the compile date and time content for assembled programs | REPORT PTF                                                                               | 641 |
| PURGE_OUT                | identifies a data set to contain the data records that result from an execution of the PURGE command | PURGE REPOSITORY                                                                       | 589 |
| QSAM                     | identifies a data set to use as a model for subsequent RUV commands         | ADD JOURNAL_MODEL  
UPDATE JOURNAL_MODEL                                                                    | 168 |
| READER_TASKS             | controls the maximum number of tasks (nn) that RUV processing uses to read archive files | SET DEFAULT  
SET  
RECOVER FORWARD                                                                    | 124  
125  
134 |
### Table 3  RUV Keywords (Part 9 of 14)

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVERY_MODE</td>
<td>specifies the Insertion Strategy for RUV to use when the RECOVER might cause inserts to the VSAM data set that create CI/CA splits</td>
<td>RECOVER FORWARD RECOVER BACKOUT</td>
<td>480 492</td>
</tr>
<tr>
<td>REGISTRATION_TIME</td>
<td>specifies a timestamp for selection of a specific backup or archive record with minimal searching of the repository</td>
<td>UPDATE ARCHIVE_FILE UPDATE BACKUP_FILE</td>
<td>265 365</td>
</tr>
<tr>
<td>REPLACE</td>
<td>specifies whether to replace an existing data set</td>
<td>ADD VSAM_NAME_MODEL UPDATE VSAM_NAME_MODEL</td>
<td>306</td>
</tr>
<tr>
<td>REPORT</td>
<td>specifies the level of report detail to display</td>
<td>SET</td>
<td>135</td>
</tr>
<tr>
<td>REPOSITORY</td>
<td>tells RUV to use the repository for the selected subsystem (or default subsystem if none was specified)</td>
<td>BACKUP REPOSITORY RESTORE REPOSITORY PRINT REPOSITORY PURGE REPOSITORY VALIDATE REPOSITORY</td>
<td>573 574 577 588 591</td>
</tr>
<tr>
<td>RESTORE_JOB_JCL</td>
<td>specify the name of the restore job JCL record that contains the JCL for restoring the backup</td>
<td>REGISTER BACKUP_FILE</td>
<td>357</td>
</tr>
<tr>
<td>RESTORE_METHOD</td>
<td>specify the name of the restore method to be registered for the backup data set</td>
<td>REGISTER BACKUP_FILE</td>
<td>359</td>
</tr>
<tr>
<td>RLS_RECOVER_PROTECTION</td>
<td>enables forward recovery of VSAM files that use record level sharing with retained locks</td>
<td>SET DEFAULT SET</td>
<td>135</td>
</tr>
<tr>
<td>RETAIN_BACKUP</td>
<td>specifies whether to keep or delete the backups after reorganization is complete</td>
<td>REORGANIZE VSAM_IN REORGANIZE VSAM_GROUP</td>
<td>442 445</td>
</tr>
<tr>
<td>RETAIN_IMAGE</td>
<td>specifies which archive images (before, after, or both) to copy</td>
<td>ARCHIVE ACCUM</td>
<td>255</td>
</tr>
<tr>
<td>RPCV_IN</td>
<td>identifies data created by the RECOVERY PLUS for CICS/VSAM (RPCV) product</td>
<td>ARCHIVE RPCV_IN</td>
<td>272</td>
</tr>
<tr>
<td>RULE_SET</td>
<td>identifies the name that is assigned to this rule</td>
<td>ADD EXTERNAL_VENDOR ADD RULE_SET ACTIVATE RULE_SET DELETE RULE_SET REPORT RULE_SET</td>
<td>184 192 198 212 420</td>
</tr>
</tbody>
</table>

---

Keywords
### Table 3  RUV Keywords (Part 10 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCRATCH</td>
<td>controls whether input archives or backups are removed from the z/OS catalog and the RUV repository</td>
<td>ARRTHE ARCHIVE ACCUM ARRTHE ARCHIVE_ARCHIVE_IN DELETE ARCHIVE_FILE DELETE BACKUP_FILE PURGE REPOSITORY</td>
<td>252</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>260</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>268</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>368</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>589</td>
</tr>
<tr>
<td>SELECTION_EXIT</td>
<td>identifies a user-written exit routine to call during RUV processing</td>
<td>RECOVER BACKOUT RECOVER FORWARD ADD VSAM_FILE ADD SEQ_FILE UPDATE SEQ_FILE UPDATE VSAM_FILE PRINT ARCHIVE_IN</td>
<td>493</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>493</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>146</td>
</tr>
<tr>
<td>SEQ_FILE</td>
<td>selects a data set for processing</td>
<td>ADD SEQ_GROUP_DEFINITION PURGE REPOSITORY REGISTER BACKUP_FILE REPORT SEQ_FILE RESTORE SEQ_FILE UPDATE SEQ_FILE</td>
<td>627</td>
</tr>
<tr>
<td>SEQ_GROUP</td>
<td>name of a sequential group</td>
<td>PURGE REPOSITORY REPORT SEQ_FILE RESTORE SEQ_FILE UPDATE SEQ_FILE</td>
<td>635</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>631</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>633</td>
</tr>
<tr>
<td>SEQ_NAME_MODEL</td>
<td>identifies a sequential name model</td>
<td>BACKUP SEQ_GROUP RESTORE SEQ_GROUP</td>
<td>629</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>631</td>
</tr>
<tr>
<td>SIMULATE</td>
<td>allows you to check for desired results before actually performing a purge</td>
<td>PURGE REPOSITORY VALIDATE REPOSITORY</td>
<td>589</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>591</td>
</tr>
<tr>
<td>SMF_IN</td>
<td>identifies the input source of SMF backup files</td>
<td>ARCHIVE</td>
<td>256</td>
</tr>
<tr>
<td>SMF_TYPE</td>
<td>specifies the SMF type and subtype</td>
<td>SET SET DEFAULT</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>136</td>
</tr>
<tr>
<td>STACKED_TAPE_COUNT</td>
<td>specifies the number of tape units available for parallel processing</td>
<td>ADD OUTPUT_MODEL</td>
<td>615</td>
</tr>
<tr>
<td>START_TIME</td>
<td>provides the beginning timestamp for the specified function</td>
<td>general information all REGISTER commands except ARCHIVE_IN DELETE ARCHIVE_FILE REPORT AUDIT REPORT SEQ_FILE REPORT VSAM_FILE RECOVER BACKOUT RECOVER FORWARD REGISTER RESTORE_JOB_JCL REGISTER RESTORE_METHOD</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>242</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>355</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>268</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>435</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>425</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>493</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>357</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>359</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
<td>Related Commands</td>
<td>Pages</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>STATUS</td>
<td>sets the active or inactive status of data sets registered in the repository</td>
<td>ADD LOG_OF_LOGS ADD LOG_STREAM ADD VSAM_NAME_MODEL all ARCHIVE commands BACKUP SEQ_GROUP BACKUP SEQ_IN BACKUP VSAM_GROUP BACKUP VSAM_IN all REGISTER commands except LOG_OF_LOGS all UPDATE commands all REORGANIZE commands</td>
<td>112</td>
</tr>
<tr>
<td>STOP_TIME</td>
<td>provides the ending timestamp for the specified function</td>
<td>DELETE ARCHIVE_FILE REPORT AUDIT REPORT SEQ_FILE REPORT VSAM_FILE RECOVER BACKOUT RECOVER FORWARD</td>
<td>113</td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>identifies the subsystem ID, ssid, of the subsystem to select for the specified function</td>
<td>SET VALIDATE SUBSYSTEM</td>
<td>125</td>
</tr>
<tr>
<td>SYNTAX_CHECK</td>
<td>scans command set for proper syntax</td>
<td>SET SET DEFAULT</td>
<td>137</td>
</tr>
<tr>
<td>SYSTEM_VARIABLE</td>
<td>requests information about all preset RUV system settings</td>
<td>REPORT SYSTEM_VARIABLES</td>
<td>178</td>
</tr>
<tr>
<td>TERM_CC</td>
<td>sets the termination condition code value that is returned at end of a job</td>
<td>SET SET DEFAULT</td>
<td>137</td>
</tr>
<tr>
<td>TERMID</td>
<td>defines the ID of the terminals to use in the range of Archive records included in the report</td>
<td>REPORT AUDIT</td>
<td>436</td>
</tr>
<tr>
<td>TIME_FORMAT</td>
<td>defines the mapping format for displaying time of day in panels and reports (it has no effect on how RUV stores a time value in the repository)</td>
<td>SET</td>
<td>138</td>
</tr>
<tr>
<td>TRACE_ON</td>
<td>activates the RUV Trace function for a specific command</td>
<td>ACTIVATE RULE_SET</td>
<td>199</td>
</tr>
</tbody>
</table>
### Table 3  RUV Keywords (Part 12 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANID</td>
<td>defines the ID of the CICS transaction to use in the range of Archive records included in the report</td>
<td>REPORT AUDIT</td>
<td>436</td>
</tr>
<tr>
<td>UNIT_OF_WORK_IN</td>
<td>identifies the input source of the archive used to recover in-flight transactions</td>
<td>ARCHIVE RECOVER COMPLETED_UNIT_OF_WORK</td>
<td>477</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>488</td>
</tr>
<tr>
<td>UNIT_OF_WORK_OUT</td>
<td>identifies the archive to contain information on un-applied in-flight transactions</td>
<td>ARCHIVE RECOVER COMPLETED_UNIT_OF_WORK</td>
<td>487</td>
</tr>
<tr>
<td>UNSELECTED_VSAM_FILES</td>
<td>writes recovery data that was not selected for an ACCUM set to a separate set of associated ARCHIVE_OUT files</td>
<td>ARCHIVE ACCUM</td>
<td>254</td>
</tr>
<tr>
<td>UPPER_CASE_TRANSLATION</td>
<td>controls whether to display data in reports in mixed case or in upper case</td>
<td>SET DEFAULT SET</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>125</td>
</tr>
<tr>
<td>USE_AUTOJOURNAL</td>
<td>controls whether RUV transforms CICS auto-journaling records into forward-recovery records</td>
<td>SET DEFAULT SET</td>
<td>139</td>
</tr>
<tr>
<td>USER_VARIABLES</td>
<td>identifies a set of user-customized variables</td>
<td>ADD RULE_SET</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD USER_VARIABLES</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE USER_VARIABLES</td>
<td>212</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT USER_VARIABLES</td>
<td>424</td>
</tr>
<tr>
<td>VSAM_FILE</td>
<td>selects a data set for processing</td>
<td>ARCHIVE ACCUM REGISTER BACKUP_FILE</td>
<td>254</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REGISTER RESTORE_JOB_JCL</td>
<td>355</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REGISTER RESTORE_METHOD</td>
<td>357</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESTORE VSAM_FILE</td>
<td>359</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT VSAM_FILE</td>
<td>458</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER BACKOUT</td>
<td>425</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER FORWARD</td>
<td>494</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD VSAM_FILE</td>
<td>473</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE VSAM_FILE</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE VSAM_FILE</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PURGE REPOSITORY</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUBMIT BACKUP</td>
<td>589</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT AUDIT</td>
<td>341</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>434</td>
</tr>
<tr>
<td>VSAM_FILE_MASK</td>
<td>defines a VSAM file name mask</td>
<td>ADD VSAM_NAME_MODEL</td>
<td>305</td>
</tr>
</tbody>
</table>
### Table 3  RUV Keywords (Part 13 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSAM_GROUP</td>
<td>name of a VSAM group</td>
<td>ARCHIVE ACCUM  DELETE VSAM_GROUP  PURGE REPOSITORY  RECOVER BACKOUT  RECOVER FORWARD  REPORT VSAM_GROUP  RESTORE VSAM_GROUP  UPDATE VSAM_GROUP  SUBMIT BACKUP  REORGANIZE VSAM_GROUP  REPORT AUDIT</td>
<td>254</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BACKUP VSAM_IN  REORGANIZE VSAM_IN</td>
<td>292</td>
</tr>
<tr>
<td>VSAM_IN</td>
<td>identifies a VSAM file to use as input for a specific command</td>
<td>BACKUP VSAM_IN  REORGANIZE VSAM_IN</td>
<td>305</td>
</tr>
<tr>
<td>VSAM_NAME_MODEL</td>
<td>identifies a VSAM name model</td>
<td>ADD VSAM_NAME_MODEL  DELETE VSAM_NAME_MODEL  REPORT VSAM_NAME_MODEL  UPDATE VSAM_NAME_MODEL  BACKUP VSAM_IN  BACKUP VSAM_GROUP  COPY VSAM_IN  COPY VSAM_GROUP  RESTORE VSAM_GROUP  RECOVER FORWARD  RECOVER BACKOUT  REORGANIZE VSAM_IN  REORGANIZE VSAM_GROUP</td>
<td>305</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BACKUP VSAM_IN  REORGANIZE VSAM_IN</td>
<td>442</td>
</tr>
<tr>
<td>WORK_DISK</td>
<td>identifies DASD</td>
<td>SET DEFAULT  SET</td>
<td>141</td>
</tr>
<tr>
<td>WORK_TAPE</td>
<td>identifies tape</td>
<td>SET DEFAULT  SET</td>
<td>141</td>
</tr>
<tr>
<td>VIEW</td>
<td>select log stream data that has been deleted</td>
<td>ARCHIVE LOG_STREAM_IN</td>
<td>231</td>
</tr>
<tr>
<td>VSAM_RULE</td>
<td>controls whether RUV should record before-images (LOG), after-images (JOURNAL), both before-images and after-images (BOTH), or no images (EXCLUDE) for the specified recoverable VSAM data set</td>
<td>ADD VSAM_SET</td>
<td>159</td>
</tr>
</tbody>
</table>
Keyword Values

One or more keyword values, enclosed by parentheses, may follow a keyword (some keywords do not have values). Values may be literals or variables. You must code literal values as shown in the documentation or models. You can code any valid values for variables. In this manual, variable values are shown in lowercase and set in italics.

Some variable values allow the use of masking (wildcard) characters. For more information, see “Using Masking (Wildcard) Characters” on page 107.

If you do not set or specify a value, RUV may do the following:

- use a program-supplied default
- produce an error message but continue running the program
- force the program to stop

Table 4 lists the variable values that are used in this manual and explains their meanings. The lengths noted in the table are the maximum length; the value may have less than the maximum length.

Table 4  RUV Keyword Variable Values (Part 1 of 2)

<table>
<thead>
<tr>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>applid8</td>
<td>8-character name associated with a VTAM application (for example, a CICS region)</td>
</tr>
<tr>
<td>comment_fields</td>
<td>four 50-character fields available for user free-form comments</td>
</tr>
<tr>
<td>ddn8</td>
<td>8-character ddname of a DD statement</td>
</tr>
<tr>
<td>devicename8</td>
<td>z/OS esoteric name used for device allocations</td>
</tr>
<tr>
<td>dsn.variables</td>
<td>data set name used in job JCL that may contain RUV system or user variables</td>
</tr>
<tr>
<td>dsn44</td>
<td>fully qualified 44-character data set name</td>
</tr>
<tr>
<td>dsn_mask44</td>
<td>44-character data set name that can contain wildcard character(s)</td>
</tr>
<tr>
<td>fileid8</td>
<td>8-character file name</td>
</tr>
</tbody>
</table>

Table 3  RUV Keywords (Part 14 of 14)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Related Commands</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSAM_SET</td>
<td>identifies the name of the rule set</td>
<td>ADD VSAM_SET ADD RULE_SET DELETE VSAM_SET REPORT VSAM_SET</td>
<td>159 192 212 430</td>
</tr>
<tr>
<td>XBMID</td>
<td>identifies XBM subsystem IDs that RUV can use for Instant Snapshot copy processing</td>
<td>SET DEFAULT SET</td>
<td>141</td>
</tr>
</tbody>
</table>
### Table 4  RUV Keyword Variable Values (Part 2 of 2)

<table>
<thead>
<tr>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>job_mask8</td>
<td>job name as referred to by z/OS JCL; can contain wildcard character(s)</td>
</tr>
<tr>
<td>job_name8</td>
<td>8-character job name</td>
</tr>
<tr>
<td>journal_model_name32</td>
<td>32-character data set name for model to be used when allocating a log file</td>
</tr>
<tr>
<td>literal</td>
<td>value inserted for test purposes only in the PARM keyword</td>
</tr>
<tr>
<td>location_name8</td>
<td>8-character field of user information that may denote location of the media</td>
</tr>
<tr>
<td>lsn26</td>
<td>fully qualified 26-character z/OS logstream name</td>
</tr>
<tr>
<td>n1, n2</td>
<td>condition codes used in SET TERM_CC command</td>
</tr>
<tr>
<td>name8</td>
<td>8-digit member name used by the Batch Journaling Facility</td>
</tr>
<tr>
<td>name32</td>
<td>32-character user-defined name</td>
</tr>
<tr>
<td>nn, nnn, mnnn, . . .</td>
<td>numeric keyword substitution values used with RUV keywords</td>
</tr>
<tr>
<td>pds.dsn.with.variables</td>
<td>name of a PDS data set that will receive JCL generated by RUV processing</td>
</tr>
<tr>
<td>ppp</td>
<td>3-digit primary space allocation</td>
</tr>
<tr>
<td>program8</td>
<td>8-character name of a user program</td>
</tr>
<tr>
<td>program_mask8</td>
<td>8-character program name that can contain wildcard character(s)</td>
</tr>
<tr>
<td>ssid</td>
<td>4-character subsystem name set during RUV installation</td>
</tr>
<tr>
<td>sss</td>
<td>3-digit secondary space allocation</td>
</tr>
<tr>
<td>ttt</td>
<td>3-character type of space being requested; can be CYL, TRK, or BLK</td>
</tr>
<tr>
<td>var_permitted</td>
<td>variable for an RUV command that may be an RUV system or user variable</td>
</tr>
<tr>
<td>variable_name</td>
<td>part of the definition of an RUV user variable that defines the variable name</td>
</tr>
<tr>
<td>variable_value</td>
<td>part of the definition of an RUV user variable that sets the variable value</td>
</tr>
<tr>
<td>&lt;vvvv&gt;</td>
<td>value that may be specified by using an RUV system or user variable</td>
</tr>
<tr>
<td>wildcards</td>
<td>wildcard symbols ? or * to facilitate a search</td>
</tr>
<tr>
<td>xxx</td>
<td>3-digit number to indicate National Language support</td>
</tr>
<tr>
<td>xxxx, . . .</td>
<td>character keyword substitution values used with RUV keywords</td>
</tr>
<tr>
<td>Xhhhhhhhhhhhhhhhhhh</td>
<td>timestamp expressed as 16 hexadecimal digits preceded by the character X</td>
</tr>
<tr>
<td></td>
<td>RUV expects all timestamps to be supplied in GMT.</td>
</tr>
<tr>
<td>yyyyjyyyyjhhmmssst</td>
<td>yyyy is the 4-digit year, jjj is the 3-digit day of the year (1 to 366), hh is the 2-digit hour of the day (00 to 23), mm is the 2-digit minutes of the hour (00 to 59), ss is the 2-digit seconds of the minute (00 to 59), t is the 1-digit tenths of the second (0 to 9)</td>
</tr>
</tbody>
</table>
Using Masking (Wildcard) Characters

Some command keyword values allow the use of the following masking (wildcard) characters:

? Use a question mark to match any single character in the position you code the question mark.

* Use an asterisk to match one or more characters starting at the position you code the asterisk.

An asterisk (* ) shown in the documentation or models for a keyword value indicates that you may supply a masking character. Masking characters are valid for REPORT and DELETE operations.

Using the COMMENT Keyword

The COMMENT keyword is available for use with many RUV commands and with many keywords that describe RUV data sets. You can use it for documentation or other purposes to help you control recovery information.

Code the COMMENT keyword as shown in the following example. For the value, you can code a maximum of 200 bytes in four 50-byte fields. Enclose each field in quotes if the field contains any spaces. Enclose all fields in parentheses.

```
COMMENT('this is a comment'
        'this is more information'
        'this is a third line of information'
        'this is the last line of the comment')
```
Using the FILTER_BY Keyword

The optional FILTER_BY keyword is available for use on the REPORT, DELETE, and UPDATE commands that process archive (ARCHIVE_FILE) and backup (BACKUP_FILE) records in the repository. You can use the keyword to filter the selection of archive or backup records by location, status, start time, and stop time.

You can use the FILTER_BY keyword to change the status, location, and comment fields in multiple records with a single UPDATE command. For example, during a disaster recovery, you can change local backups and archives to inactive status and remote backups and archives to active status.

Because the syntax of the FILTER_BY keyword is identical on REPORT, DELETE, and UPDATE commands, you can run a REPORT command to see which archive and backup records would be deleted or updated before physically deleting or updating the records.

Code the FILTER_BY keyword with the LOCATION, STATUS, START_TIME, and STOP_TIME keywords as shown in the following syntax for the UPDATE ARCHIVE_FILE command:

```plaintext
UPDATE ARCHIVE_FILE(ddn_or_dsn, *, ...) 
{ FILTER_BY(
    { LOCATION(location_name8,*), ..) } 
    { STATUS(ACTIVE | INACTIVE) } 
    { START_TIME(yyyyjjjhhmmss) } 
    { STOP_TIME(yyyyjjjhhmmss) } 
) 
{ LOCATION(location_name8) } 
{ STATUS(ACTIVE | INACTIVE) } 
{ COMMENT(comment_fields) }
```

RUV reads each archive (or backup) record in the repository and qualifies the record for update processing based on the criteria you provide with the FILTER_BY keyword. If the record meets the FILTER_BY criteria, RUV updates it according to the values that you specify with the LOCATION, STATUS, and COMMENT keywords.

If you specify a start time, the filter selects records with timestamps that are equal to or later than the start time. If you specify a stop time, the filter selects records with timestamps that are equal to or earlier than the stop time. Therefore, you can specify a start time and stop time pair to select a time range.

For more information about these keyword values, see “Using the LOCATION Keyword” on page 110, “Using the STATUS Keyword” on page 112, “Using the START_TIME Keyword” on page 111, and “Using the RUV ISPF Interface” on page 113.
FILTER_BY Keyword Example

The following example shows how the FILTER_BY keyword can help with the mass status update of archive and backup records in a disaster recovery situation. The following set of UPDATE commands resets the status of the LOCAL backups and archives to inactive:

```
UPDATE ARCHIVE_FILE(*)
  FILTER BY(LOCATION(LOCAL))
  STATUS(INACTIVE);
UPDATE BACKUP_FILE(*)
  FILTER BY(LOCATION(LOCAL))
  STATUS(INACTIVE);
```

The following set of UPDATE commands resets the status of REMOTE backups and archives to active:

```
UPDATE ARCHIVE_FILE(*)
  FILTER BY(LOCATION(REMOTE))
  STATUS(ACTIVE);
UPDATE BACKUP_FILE(*)
  FILTER BY(LOCATION(REMOTE))
  STATUS(ACTIVE);
```

Using the HOSTCOPYMODE Keyword

The HOSTCOPYMODE keyword is valid on ADD VSAM_NAME_MODEL and UPDATE VSAM_NAME_MODEL commands. You can use this keyword to specify the host copy mode parameter for Instant Snapshot copies. The following values are valid:

- If you want the task to share control of the data set that is being copied, specify HOSTCOPYMODE(SHARED).
- If you want the task to obtain exclusive control of the data set that is being copied, specify HOSTCOPYMODE(EXCLUSIVE). The task fails if it cannot obtain exclusive control.

The default value is HOSTCOPYMODE(SHARED).
Using the LOCATION Keyword

The optional LOCATION keyword is available for use on many commands. You can use it to identify, in eight bytes or less, the physical destination of the selected data. The value (location_name8) is a free-form format and has no RUV-internal use.

It is recommended that, if you use this keyword, you establish a standard convention for managing off-site data files.

If you omit this keyword, the location field for the selected data is blank.

---

**NOTE**
The LOCATION keyword can be used as a reporting filter in batch or TSO processing.

---

Using the MUST_COMPLETE Keyword

You can use the optional MUST_COMPLETE keyword to control whether the archive process for a specified ARCHIVE_OUT data set must complete successfully. Specify one of the following values:

**YES**  Abort processing if the data transfer cannot be completed to the specified data set. This value is the default.

**NO**   Allow archive processing to continue even if the archive receives an X37 or other abend. Processing stops if no ARCHIVE_OUT data set can be produced.

Using the STOP_TIME Keyword

The STOP_TIME keyword is available on some commands to specify the stop (latest) timestamp of the range of journal records to select for processing.

You can specify the value of the STOP_TIME keyword with the same formats that are valid for the START_TIME keyword. The default value is high values (X'FFFFFFFFFFFFFFFF') or 2042.015.23:53:47
Using the START_TIME Keyword

The START_TIME keyword is available on some commands to specify the start (earliest) timestamp of the range of records to select for processing.

RUV assumes that you are supplying all timestamps in GMT. To determine the timestamp associated with BACKUP or an ARCHIVE file, run a REPORT command. This will assist you in selecting the GMT date which RUV recorded.

As shown in the following examples, you can specify the value of the START_TIME keyword in several formats. The earliest valid time is 1980. The default value is low values (X'0000000000000000').

- without separators

```
START_TIME(20092192014)
```

- with separators between each component value

```
START_TIME(2009.219/20:14:00)
START_TIME(2009219/20:14)
START_TIME(2009.219.20.14.00)
```

You can use a slash (/), a period (.), or a colon (:) as a separator. No character (including spaces) other than a slash, period, or colon may separate time values.

You can use separators instead of a value to use the default value for that component value. Starting from the left, a separator character by itself specifies the default value for that position. For example, you can specify START_TIME(/18) to specify 6:00 PM of the current day and current year. You can specify START_TIME(2009.1) to specify January 1, year 2009, hour 00:00:00.

- as hexadecimal store clock format

Specify Xhhhhhhhhhhhhhhhhhh (a literal X followed by 16 hexadecimal digits) to supply a time in hexadecimal format.
Using the STATUS Keyword

The optional STATUS keyword is available on many commands. Use it to specify the status of a data set. RUV automated processes use active records only. Only one file of each type for each VSAM data set should have an active status. You can use the UPDATE command to change the status of the data set. You can display the current status by using the REPORT command.

Specify one of the following values. The default value is ACTIVE.

**ACTIVE**

Register the data set as active in the repository. If it is an ARCHIVE_OUT data set, during the consolidated accumulation change the accumulated pieces of the processed ARCHIVE_IN data sets to inactive status. This action causes the accumulated ARCHIVE_OUT data sets to be used for subsequent recovery processing, but the original archives (that were used for input to the accumulation) are not used.

---

**NOTE**

During consolidated accumulation, only the VSAM files that are actually accumulated are marked as inactive in the repository for the input archives in the repository. The entire input archive is marked as inactive only when all files on it are inactive.

If multiple ARCHIVE_OUT data sets are created and marked as active, RUV processes all of them for recovery, even if they are copies of each other.

---

**NOTE**

For performance reasons, you should not mark duplicate data as active.

---

**INACTIVE**

Mark the data set as inactive in the repository. If you specify STATUS(INACTIVE) for an ARCHIVE_OUT data set, input archive status remains active. RUV does not use inactive archives in recovery processing or inactive backups during restore or recovery processing.
The RUV ISPF interface is an automated recovery tool that can guide you through the recovery process and provide information for successful recovery of VSAM files. VSAM data recovery is an infrequent event. The design of the interface allows people who have little or no experience in using interface facilities to process a recovery.

Starting the RUV ISPF Interface

To start the RUV ISPF interface, perform the following steps:

1. From a TSO session, issue one of the following commands:

   TSO RUVISPF
   TSO EXEC 'yourname.DLIB(RUVISPF)'

   If you are running virtual I/O (VIO) for temporary data sets, use the following amended EXEC command:

   TSO EXEC 'yourname.DLIB(RUVISPFZ)'

2. Specify an active subsystem.

   If more than one subsystem is available, the RUV Options panel with the Subsystem List popup window (Figure 18) is displayed. Type any character to the left of the name of the subsystem that you want to use, and press Enter. If only one subsystem is available, the popup window is not displayed, and RUV uses the available subsystem.
Figure 18  Subsystem List Pop-up Window

The RUV Options panel (Figure 19) is now ready to use.
Using the RUV Options Panel

Use the RUV Options panel (Figure 19) to choose the function that you want to perform.

All areas on the RUV Options panel, except the command line, are enabled for point-and-click processing. You can select an action bar choice, the Subsystem ID field, or a menu choice by placing the cursor on a field and pressing Enter. You can double-click a mouse button that you have assigned to the Enter key function if you are operating RUV from a PC workstation.

The action bar is available at the top of every ISPF interface panel. The choices that are displayed on the action bar depend on the panel and on the view mode. For more information, see “Using Action Bar Choices” on page 117.

1 In the Subsystem ID field, type or verify the subsystem ID of the subsystem to which you want to connect.

Normally, you select the subsystem that you want to use when you start the ISPF interface. To change the current subsystem connection, you can enter a new subsystem ID in this field and press Enter.
2 Select an option by typing the number of the option in the choice entry field. The following options are available:

1 Perform a VSAM recovery. For more information, see “Using the ISPF Interface for Recovery” on page 495.

2 Display all sequential data set names or VSAM base cluster and path data set names. For more information, see “VSAM DSN List” on page 376.

3 Display APPLID/FILEID pairs. For more information, see “Applid/Fileid List” on page 378.

4 Display RUV archive data set names and related detail information. For more information, see “Archive DSN List” on page 387.

5 Display information related to registered VSAM backups. For more information, see “Backup File List” on page 389.

6 Display a list of log stream names. For more information, see “Log of Logs List” on page 391.

7 Display Group and Output Model and VSAM_NAME models. For more information, see “Groups / Output / VSAM Models” on page 393.

8 Display operating system version information and activity details for subsystems, and display rule sets. For more information, see “Using the ISPF Interface to Work with Rule Sets” on page 205.

9 Display message information by looking up a message number or by looking up the text. For more information, see “Message Lookup” on page 398.

10 Generate job JCL records for performing external backup and restore processes. For more information, see “Working with Job JCL Records for Backup and Restore Processes” on page 327.

11 Monitor RUV jobs that are executing on this operating system image. For more information, see “RUV Job Monitoring” on page 400.
Using Action Bar Choices

The action bar is available at the top of every ISPF interface panel. The choices that are displayed on the action bar depend on the panel and on the view mode. The following sections describe standard action bar choices.

Using File Action Bar Choices

The File action bar choice contains pull-down choices that are appropriate for the panel on which it is displayed. Pull-down choices may not be available under certain circumstances. The Delete choice, for instance, is not available if you are in inquiry mode.

The following standard pull-down choices may be available from the File action bar choice:

Exit
  Go back to the previous panel.

Exit RUV
  Leave the RUV ISPF interface, and return to the point from which you started the interface.

Delete
  Remove the selected record from the repository.

Change Status
  Toggle the selected record from active to inactive status or from inactive to active status.

Using View Action Bar Choices

The View action bar choice contains pull-down choices that are appropriate for the panel on which it is displayed.

Inquiry Mode, Update Mode, and Command Mode choices toggle among the three available viewing modes: command, update, and inquiry. The currently active mode is not available as a choice. Certain actions, such as Delete and Update, are not available if you are in inquiry mode.

The following standard pull-down choices may be available from the View action bar choice:

Inquiry Mode
  View the information only. You cannot change any information in the repository. You cannot issue any commands.
Using Action Bar Choices

**Update Mode**
View and change information in the repository. You cannot issue any commands.

**Command Mode**
View and change information in the repository. You can issue subsystem commands. For more information about commands, see Chapter 13, “Working with BMC Software Subsystems.”

**Filter**
Use this choice to limit the entries that are displayed on a table display panel. For more information, see “Using Display Filters” on page 119.

**Flip View**
Filter (hide) the entries that are currently being displayed on a table display panel, and display the entries currently being filtered (hidden).

**Refresh**
Update the current panel with panel information that has been added or updated since you displayed the current view of the panel.

**Using Tools Action Bar Choices**

The Tools action bar choice includes the following choices:

**Date Conversion**
Convert Julian dates to Gregorian dates and display a range of converted dates in a popup window. For more information, see “Using the Date Conversion Tool” on page 120.

**TOD Conversion**
Convert a double-word store clock date in hexadecimal format to your preferred date and time format. For more information, see “Using the TOD Conversion Tool” on page 121.

**User Preferences**
Select view mode, language, date and time formats, and miscellaneous preferences. For more information, see “Setting User Preferences” on page 122.
Using Display Filters

You can use the Filter choice from the View action bar choice to limit the entries displayed on a table display panel. When you select this choice, RUV displays the Display Filters popup window (Figure 20).

Figure 20 Display Filters Popup Window

In each field, you can type a mask that RUV uses when selecting records. You can use the masking (wildcard) characters ? and * in the mask; ? matches any single character in the position, and * matches any number of characters in the position. If the field contains * only, the mask matches all characters, causing RUV to ignore the field as a filter.

When you exit from the Display Filters popup, the table is redisplayed with only the entries that match your filter.
Using the Date Conversion Tool

When you select the Date Conversion choice from the Tools action bar choice, RUV displays the Date Conversion popup window (Figure 21). Use this tool to convert Julian dates to Gregorian dates and display a range of converted dates.

**Figure 21  Date Conversion Popup Window**

At entry, the table is positioned with the current date in the middle of the list. You can page forward and backward with the PAGE DOWN and PAGE UP commands. You can type over the year or Julian date field to display the Gregorian date.
Using the TOD Conversion Tool

When you select the **TOD Conversion** choice from the **Tools** action bar choice, RUV displays the TOD Clock Date/Time Conversion popup window (Figure 22). Use this tool to display and convert a double-word store clock date in hexadecimal format to your preferred date and time format.

**Figure 22    TOD Clock Date/Time Conversion Popup Window**

At entry, the TOD clock value field contains the current GMT date. You can type over the clock value field to display the date in your preferred format. (You set your preferred format with the **User Preferences** choice from the **Tools** action bar choice.)
Setting User Preferences

When you select the User Preferences choice from the Tools action bar choice, RUV displays the User Preferences panel (Figure 23). Use this panel to select view mode; language, date and time formats; and miscellaneous preferences.

Enter the values to use for each field. For more information about any field, place the cursor in the field and press F1.

Figure 23  User Preferences Panel
Using RUV Subsystem Console Commands

RUV uses the BMC Software Primary subsystem (BMCP) and the BMC Software Consolidated subsystem (BCSS) to perform various system-level activities. You can use the commands listed in Table 5 to work with these subsystems. You can enter the commands from the system console or a TSO SDSF session. The commands for BCSS are applicable to both the public and private copies of the subsystem. You must enter the subsystem ID (shown in the table as ssid) for BCSS commands.

For more information about the subsystems and commands, see Chapter 13, “Working with BMC Software Subsystems.”

Table 5  RUV Subsystem Console Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>/BMCP STATUS</td>
<td>displays the status of the BMC Primary subsystem (BMCP)</td>
<td>541</td>
</tr>
<tr>
<td>/BMCP SHUTDOWN</td>
<td>shuts down the BMCP in an orderly manner by terminating the BMCP address space and BMCP command processing</td>
<td>542</td>
</tr>
<tr>
<td>/ssid STATUS</td>
<td>displays the status of the specified BCSS</td>
<td>549</td>
</tr>
<tr>
<td>/ssid SHUTDOWN</td>
<td>shuts down the specified BCSS in an orderly manner by terminating the BCSS address space and all components within it</td>
<td>550</td>
</tr>
<tr>
<td>/ssid REINIT RUV</td>
<td>initializes the RUV subsystem interface</td>
<td>548</td>
</tr>
<tr>
<td>/ssid RUV TAKEOVER BWOCOORD</td>
<td>changes ownership of the Backup-While-Open (BWO) coordinator to the specified BCSS</td>
<td>553</td>
</tr>
<tr>
<td>/ssid RUV DEBUG BWODELNT</td>
<td>provides internal diagnostic information</td>
<td>553</td>
</tr>
<tr>
<td>/ssid RUV HELP</td>
<td>displays command formats for RUV commands</td>
<td>550</td>
</tr>
<tr>
<td>/ssid RUV QUIESCE</td>
<td>stops activity in RUV in an orderly manner for the specified scope</td>
<td>551</td>
</tr>
<tr>
<td>/ssid RUV ENABLE</td>
<td>enables activity in RUV for the specified scope</td>
<td>550</td>
</tr>
<tr>
<td>/ssid RUV DISABLE</td>
<td>stops activity in RUV immediately for the specified scope</td>
<td>552</td>
</tr>
<tr>
<td>/ssid RUV STATUS</td>
<td>displays the status of RUV for the specified scope</td>
<td>552</td>
</tr>
<tr>
<td>/ssid RUV ACTIVATE RULE_SET(name32)</td>
<td>activates a name32 rule set</td>
<td>549</td>
</tr>
</tbody>
</table>
Setting Default and Override Environment Options

RUV allows you to control many of the options related to the RUV environment. You use the SET DEFAULT command to establish system default values. You use the SET command to establish option values at the job (local) level.

Establishing System and Local Job Options

The default settings are retained in the subsystem’s repository and take effect when the subsystem is selected by the first set statement in the RUV utility program RUVZSM0. If the SET DEFAULT command is not the first control statement, RUV obtains default values from the publicly enabled RUV subsystem.

You can override default settings at the job (local) level by coding option values with the SET command for the job.

Using the SET DEFAULT Command

To set default environmental options for a subsystem, use the SET DEFAULT command, as shown in the following example. The DEFAULT keyword indicates that this command applies to all subsequent RUV processes that use this subsystem. Other keywords on the command correspond to options you can control.

Figure 24 SET DEFAULT Command Example (Part 1 of 2)

```
SET DEFAULT
  ASSOCIATED_ADDRESS_SPACE_TASKS(10)
  BACKUP_CHANGED_ONLY(NO)
  BACKUP_METHOD(RUV BWO SNAPSHOT SMS EXTERNAL)
  CYCLES(3)
  DATE_FORMAT(YYYY.JJJ)
  DAYS(45)
  ESOTERIC(SYSALL)
  GAP (STOP)
  INCREMENTAL_BACKUP_SUPPORT(YES)
  LANGUAGE (ENU)
  LINE_COUNT(56)
  PAGE_WIDTH (120)
  READER_TASKS (100)
  RECOVERY_MODE(NIS)
  RLS_RECOVER_PROTECTION(NO)
  SMF_TYPE (0,0)
```
Using the SET Command

To set environmental options for a job, use the SET command, as shown in the following example. Keywords on the command correspond to options you can control.

**Figure 25  SET Command Example**

```plaintext
SET
ASSOCIATED_ADDRESS_SPACE_TASKS(10)
BACKUP_CHANGED_ONLY(NO)
BACKUP_METHOD(RUV BWO SNAPSHOT SMS EXTERNAL)
CYCLES(3)
DATE_FORMAT(YYYYJJJ)
DAYS(45)
ESOTERIC(SYSALL)
GAP(STOP)
INCREMENTAL_BACKUP_SUPPORT(YES)
LANGUAGE(ENU)
LINE_COUNT(56)
MAX_CC(08)
PAGE_WIDTH(120)
<+1>(3)
READER_TASKS(100)
RECOVERY_MODE(NIS)
REPORT(DETAIL)
RLS_RECOVER_PROTECTION(YES)
SMF_TYPE(1,2)
SUBSYSTEM(RUVT)
SYNTAX_CHECK(YES)
TERM_CC(0,0)
TIME_FORMAT(HH:MM:SS)
USE_AUTOJOURNAL(NO)
UPPER_CASE_TRANSLATION(OFF)
WORK_DISK(SYSALLDA)
WORK_TAPE(SYSALLDA)
XBMID(XBM1, XBM2)
;
```
Using the ASSOCIATED_ADDRESS_SPACE_TASKS Keyword

Use the ASSOCIATED_ADDRESS_SPACE_TASKS keyword to determine how many address spaces will be attached during recovery processing. Associated address spaces are used to perform alternate index builds in parallel. Set the value nn to specify the number of address spaces that will be attached during recovery processing.

The nn value may be any number from 0 to 200.

Using the BACKUP_CHANGED_ONLY Keyword

Use the BACKUP_CHANGED_ONLY keyword to back up files within a VSAM Group that have been changed. This keyword allows you to obtain a partial backup of a group’s contents without backing up all files in a group.

Set one of the following values. The default value is NO.

YES A backup will be taken of only those files within a group that have changed.

NO A complete backup of all group files will be taken regardless of whether they have changed.

Using the BACKUP_METHOD Keyword

RUV can perform backups and restores with any of several methods. You can use the BACKUP_METHOD keyword to control how these backup methods are used during the backup process and restore process.

NOTE

The BACKUP_METHOD keyword simply enables or disables use of a backup or restore method. Other commands in the command set invoke the backup or restore process.

Use the BACKUP_METHOD keyword on the SET or SET DEFAULT command to control the use of backup methods during the backup process, restore process, or both (depending on the commands that follow the SET or SET DEFAULT command). You can code multiple values for the BACKUP_METHOD keyword, but do not code a disabling value with its corresponding enabling value. Set one of the following values. The default is BACKUP_METHOD(RUV BWO SNAPSHOT SMS EXTERNAL).
To disable a backup method, you can use NO_BWO, NO_SNAPSHOT, NO_SMS, or NO_EXTERNAL.

**NOTE**
When RUV selects the most current backup, the value of the BACKUP_METHOD keyword determines the backups that are considered usable.

RUV For the backup process, enable use of the RUV backup method. For the restore process, enable the use of backups that were produced by the RUV or RUV backup method.

BWO
For the backup process, enable use of the Backup-While-Open (BWO) feature and the Record Level Sharing (RLS) feature to produce fuzzy backups. For the restore process, enable the use of fuzzy backups that were produced by the BWO or RLS feature. A fuzzy backup is not complete by itself; during a recovery, you must also use journal data to achieve data integrity.

NO_BWO
For the backup process, disable use of the BWO feature and the RLS feature. (Do not produce fuzzy backups.) For the restore process, disable the use of fuzzy backups that were produced by the BWO feature or RLS feature.

BACKUP_REPLY
If the Backup function cannot obtain exclusive ownership of a data set, issue the following message to the master console:

`RUv204105A RUv <dsname> Backup Integrity exposure. REPLY GO, RETRY, or CANCEL.`

The operator can reply to the message with one of the following responses:

- **GO** — Skip the backup of the current file and continue to the backup of the next file.
- **RETRY** — Attempt to obtain exclusive control of the current file.
- **CANCEL** — Terminate the backup job.

**NOTE**
A value of RUV with this keyword is supported for downward compatibility purposes only.

When RUV selects the most current backup, the value of the BACKUP_METHOD keyword determines the backups that are considered usable.
SNAPSHOT
For the backup process, enable use of the BMC Software SNAPSHOT UPGRADE FEATURE (SUF) for VSAM component to produce Instant Snapshot copies. You can also make an Instant Snapshot copy of a sequential file. (You must specify other keywords and enable other components to produce Instant Snapshot copies, as explained in “Creating Instant Snapshot Copies” on page 302.) For the restore process, enable use of SUF to restore an Instant Snapshot copy by using the features of the intelligent storage device.

NOTE
To use SUF with RUV to produce Instant Snapshot copies, you must install and configure the EXTENDED BUFFER MANAGER (XBM) subsystem and a supported intelligent storage device.

NO_SNAPSHOT
For the backup process, disable use of SUF to produce Instant Snapshot copies. This value allows you to turn off SUF without making other keyword changes or shutting down the XBM subsystem. For the restore process, disable use of SUF to restore an Instant Snapshot copy; the file will be restored with a VSAM-to-VSAM or sequential-to-sequential file copy process instead. You can set this value if you want the file to be physically reorganized during the restore process.

SMS
For the restore and recovery process, enable use of backups that are produced and managed by Storage Management Subsystem (SMS). RUV does not invoke SMS for the backup process.

NO_SMS
For the restore and recovery process, disable use of SMS backups. RUV does not invoke SMS for the backup process.

EXTERNAL
For the restore and recovery process, enable use of backups that are produced with RUV external backup support. This value has no effect on the external backup process.

NO_EXTERNAL
For the restore and recovery process, disable use of backups that were produced with RUV external backup support. This value has no effect on the external backup process.
Using the CYCLES Keyword

Use the CYCLES keyword on the SET DEFAULT or SET command to control the number of iterations (cycles) of historical data to retain in the repository. Normally, this cycle includes the backup data and the archive data. The cycle begins when a backup is created for a VSAM or sequential file. All archive data created after this backup becomes part of this cycle. The next backup logically begins a new cycle.

Set the value of the CYCLES keyword to the number (0 to 999) of cycles you want to retain. RUV maintains the information about the cycles and their related data in the RUV repository. Setting a large number of cycles increases the number of data records in the repository and adds overhead to any processing that uses the repository. You should select a cycle number that protects your data and allows for any historical recovery requirements.

As new cycles are added to the repository, previous cycles are no longer needed. The data associated with old cycles remains in the repository until the PURGE REPOSITORY command is executed. This command removes the oldest cycle data from the repository until the number of cycles you have set with the CYCLES keyword remain.

If you code the CYCLES keyword, do not code the DAYS keyword.

Using the DATE_FORMAT Keyword

Use the DATE_FORMAT keyword on the SET DEFAULT or SET command to control the format of dates that RUV displays in reports and panels. Use one of the following values. The default value is YYYY.JJJ.

**YYYY.JJJ**
Display dates in Julian format. For example, September 27, 2009, is displayed as 2009.270.

**MM/DD/YYYY**
Display dates in month/day/year format. For example, September 27, 2009, is displayed as 09/27/2009.

**DD/MM/YYYY**
Display dates in day/month/year format. For example, September 27, 2009, is displayed as 27/09/2009.

**YYYY/MM/DD**
Display dates in year/month/day format. For example, September 27, 2009, is displayed as 2009/09/27.
USER_DEFINED

Display dates in your own customized date format, as explained in the following section.

Using a User-Defined Date Format

Perform the following steps if you want to create and use your own customized date format. RUV will use the customized date format when displaying dates in reports and panels.

1. Copy or rename the RUVMENUF load module for a backup to use (in case an error occurs in the following steps).

2. Copy source member RUVMENUF from the HLQ.SOURCE library to a user library.

3. Locate the string “DFLDATEO” in the RUVMENUF source.

4. Arrange the string “TEXT=....” to the date format you want to use.

5. Assemble the module and include the HLQ.SOURCE library in the SYSLIB concatenation.

6. Link the changed module into your RUV load library, replacing the original RUVMENUF module.

7. RUN the RUVZSM0 utility with the SET DATE_FORMAT (USER_DEFINED) keyword.

8. Check the report output for the new date format.
Using the DAYS Keyword

Use the DAYS keyword on the SET DEFAULT or SET command to control the number of 24-hour days of historical data to retain in the repository.

The primary difference between using the DAYS and CYCLES keywords is that DAYS is an absolute number of 24-hour days, while CYCLES is a number of groups of data files (cycles) that contain a complete recoverable VSAM or sequential file.

Set the value of the DAYS keyword to the number (0 to 999) of days you want to retain.

**NOTE**

The REGISTER LOG_OF_LOGS command does not use the default DAYS value that is recorded in the repository. When you specify the DAYS keyword for the REGISTER LOG_OF_LOGS command, the DAYS keyword applies to that execution only.

**WARNING**

If you code the DAYS keyword, do not code the CYCLES keyword.

Using the ESOTERIC Keyword

Use the ESOTERIC keyword on the SET DEFAULT or SET command to define an esoteric device that you want to use for Instant Snapshot copy processing. If you set this keyword, RUV attempts to use one of the available volumes in the pool of esoteric devices. The keyword value is the defined name of the esoteric device.

Using the GAP Keyword

Use the GAP keyword on the SET DEFAULT or SET command or the ARCHIVE command to control the action to be taken when RUV detects missing data. Missing data is specific to log stream processing and is defined as a gap.

When RUV is processing a log stream, it can get return codes from the z/OS logger that indicate a gap. In most cases, the cause of the missing data becomes apparent when you review all processes that occurred on that log stream. After RUV detects a gap (missing data), RUV can stop the process or continue.
Using the INCREMENTAL_BACKUP_SUPPORT Keyword

Set one of the following values. The default value is STOP.

**STOP**
Stop processing the log stream. It is recommended that you use the default. After you analyze the cause of the gap and determine that the process needs to continue, you can rerun the process with the WARN or IGNORE value.

**WARN**
Produce a message, generate return code 4, and continue processing.

**IGNORE**
Produce a message, generate return code 0, and continue processing.

Using the INCREMENTAL_BACKUP_SUPPORT Keyword

Use the INCREMENTAL_BACKUP_SUPPORT keyword to prevent non-RUV backup programs from creating duplicate backups. When RUV backs up a data set and the INCREMENTAL_BACKUP_SUPPORT keyword is set to YES (or defaulted to YES), the data-set-changed indicator in the VTOC is turned off. When the data-set-changed indicator in the VTOC is turned off, SMS and other backup programs will not create a backup.

RUV backups need to be run in front of your existing incremental backups in order for incremental backups to be effective.

Set one of the following values. The default value is YES.

**YES** The VTOC indicator is turned off to prevent backups by SMS or other backup programs that honor the indicator.

**NO** The VTOC indicator is unchanged and a backup may be performed by SMS or other backup programs.

Using the LANGUAGE Keyword

The LANGUAGE keyword on the SET DEFAULT or SET command is intended for future support of national languages. For the current release of RUV, English is the only supported language. The keyword value is ENU (English, United States).
Using the **LINE_COUNT** Keyword

Use the LINE_COUNT keyword on the SET DEFAULT or SET command to control the number of lines to be printed per page. Set the value to meet your needs for page sizes that range from 15 to 9999999 lines. The default value is 56.

Using the **MAX_CC** Keyword

Use the MAX_CC keyword to set a maximum two-digit condition code.

Using the **PAGE_WIDTH** Keyword

Use the PAGE_WIDTH keyword on the SET DEFAULT or SET command to control the width of the report page. Set the value to a two- or three-digit number in the range from 30 to 121 to indicate the number of characters to print per line. The default value is 121. If you set this keyword to a value that is smaller than the displayed line, printed data will wrap to the next line.

Using the **PASSWORD_EXPIRATION_CC** Keyword

Use the PASSWORD_EXPIRATION_CC keyword on the SET DEFAULT or SET command to control the condition code that RUV issues during the warning period that precedes the expiration of the product authorization password. Set the value to the condition code (0 or 4). The default value is 4.

Using the **<+1>** Keyword (for X37 Support)

The <+1> variable is used for X37 support during batch journaling. Use the <+1> keyword on the SET command to correct the value of the <+1> variable after restoring the RUV repository from a backup. For more information, see “Using the <+1> Variable for X37 Support” on page 177.

The current value of the <+1> variable is stored in the RUV repository. A repository restore process causes the value of the <+1> variable to be reset to the value it had at the time of the repository backup. You can then change the value of the <+1> variable by executing the following control statement:

```
SET <+1>(nn);
```
The value \( nn \) is the value that the \(<+1>\) variable had at the time immediately before the restore process. After the command executed, the next usage of the \(<+1>\) variable generates a value of \( nn + 1 \).

The \( nn \) may be any value from 1 to 9,999,999.

---

**Using the READER_TASKS Keyword**

Use the READER_TASKS keyword on the SET DEFAULT or SET command to control the maximum number of reader task control blocks (TCBs) that RUV can attempt to attach for reading input files. This option is used in situations where RUV can process input files simultaneously and you want to limit the number of input tape drives. The resources that you make available with this keyword can also prevent multitasking of the reader process.

The \( nn \) number can be any value from 1 to a maximum of 99 (100) reader tasks. The default value is 99. See “Parallel Processes” on page 54 for additional information about reader tasks.

---

**Using the RECOVERY_MODE Keyword**

Use the RECOVERY_MODE keyword on the SET DEFAULT or SET command to control the Insertion Strategy for RUV to use when the RECOVER might cause inserts to the VSAM data set that create CI/CA splits. This keyword allows you to control the Insertion Strategy.

The default is NIS (Normal Insertion Strategy), which is good for most files. Files that have a large number of records inserted at one spot might benefit from the RECOVERY_MODE being set to SIS (Sequential Insertion Strategy). Before changing this parameter setting, see the Warning on page 475.
Using the REPORT Keyword

Use the REPORT keyword on the SET command to set the level of report detail to generate. Each command that writes data to the SYSPRINT DD statement may produce these varying levels of detail. Only some commands will produce additional levels of detail.

Set one of the following values. The default value is SUMMARY.

**SUMMARY**

- Produce minimum data, such as record counts and data set names.

**DETAIL**

- Produce both SUMMARY data and additional lines that show in-depth control counts and status.

**FULL**

- Produce both SUMMARY and DETAIL information and hexadecimal display of certain user data. Normally, you should use this value only for problem analysis because it can produce large amounts of data.

Using the RLS_RECOVER_PROTECTION Keyword

Use the RLS_RECOVER_PROTECTION keyword on the SET DEFAULT or SET command to enable RUV to perform a forward recovery of VSAM files that use the VSAM Record Level Sharing (RLS) feature with retained locks that are held for individual records. If RLS recovery protection is enabled, the only change to your normal recovery procedure is to quiesce any RLS files that are open to CICS. Before the RUV recovery job starts, perform the quiesce with the following command:

```
CEMT SET DSN(data set name) QUIESCED
```

To perform a forward recovery of an RLS file that has retained locks, RUV issues standard subcommands to the sharing control data set (SHCDS). The SHCDS contains information that is required for DFSMS/MVS to maintain data integrity in an RLS environment when failures occur. RUV issues the following SHCDS subcommands during the recovery of RLS files:

- FRSETRR
- FRUNBIND
- FRBIND
- FRRESETRR
Using the SMF_TYPE Keyword

For the RLS_RECOVER_PROTECTION keyword, set one of the following values. The default value is NO.

- **NO**  Disable SHCDS commands during forward recovery of RLS files with retained locks.
- **YES**  Enable forward recovery of RLS files with retained locks.

**NOTE**

Far in advance of a possible recovery, set the RLS_RECOVER_PROTECTION keyword to YES and use the STGADMIN.IGWSHCDS.REPAIR facility to establish UPDATE authority with your security package (such as RACF or ACF2). For detailed information, refer to the RACF authorization table in the IBM document *DFSMS/MVS: Access Method Services for the Integrated Catalog Facility*.

For the RLS_RECOVER_PROTECTION keyword, set one of the following values. The default value is NO.

- **NO**  Disable SHCDS commands during forward recovery of RLS files with retained locks.
- **YES**  Enable forward recovery of RLS files with retained locks.

**Using the SMF_TYPE Keyword**

Use the SMF_TYPE keyword to indicate the SMF type number. The values mean the following:

- $x=$SMF type number, $y=$SMF subtype number
- Valid $x$ values are 0 or 128 through 255
  - If $x$ is set to zero, SMF recovery information will not be recorded.
  - If $x$ is set to 128 through 255, SMF recovery information will be recorded.
- Valid $y$ values are 0 or 1 through 10. The $y$ value does not have to be set.

**Using the SUBSYSTEM Keyword**

Use the SUBSYSTEM keyword on the SET command to select the BCSS to use for the job. Set the value to the subsystem ID (ssid) that is assigned to the subsystem you want to use. The subsystem ID is set during installation of the RUV product.

For more information, see Chapter 13, “Working with BMC Software Subsystems.”
Using the **SYNTAX_CHECK** Keyword

Use the **SYNTAX_CHECK** keyword to place the RUV command processor into a syntax checking mode. Statements following the command are checked for syntax accuracy but not executed. You cannot exit syntax checking mode after it has been entered.

Syntax checking mode allows you to validate command statements easily. Only the syntax of a command is checked. The validation of data (valid DD names, data set names, values) is not checked.

Set one of the following values. The default value is **YES**.

**YES** The RUV command processor is placed into a syntax checking mode.

**NO** The RUV command processor is not placed into a syntax checking mode.

Using the **TERM_CC** Keyword

Use the **TERM_CC** keyword on the SET DEFAULT or SET command to control the value of the termination condition code that is returned at end of a job.

You can use this option to force RUV to continue execution after receiving a termination condition code. A termination condition code is any code equal to or greater than 8. For example, you can use this option to force a report command to be processed even when the previous command fails, assuring that the report will always be produced.

**NOTE**

You should not force execution of commands that have nonrecoverable side effects.

Code the value of the **TERM_CC** keyword as \((n1, n2)\). The value \(n1\) controls RUV execution. At completion of each command execution, RUV compares its internal condition code to the \(n1\) value. If the \(n1\) value is equal to or greater than the internal condition code, RUV will proceed to the next command; otherwise it will stop processing. The default value for \(n1\) is 4.

The \(n2\) value is used to decide when 0 is returned to z/OS. At the completion of each command execution, RUV compares its internal condition code to the \(n2\) value. If \(n2\) is greater than or equal to the internal condition code, RUV presents 0 to z/OS. Otherwise, it presents the highest condition code encountered. The default value for \(n2\) is 0.
Using the **TIME_FORMAT** Keyword

For example, in the following command, n1 allows the command execution to continue even if a step received a condition code 4 (a warning); n2 will present a zero (0) to z/OS as the step completion code unless some command returned a code greater than 4:

```
SET TERM_CC(4,4);
```

**Using the **TIME_FORMAT** Keyword**

Use the **TIME_FORMAT** keyword on the SET DEFAULT or SET command to control the layout structure of time notations that RUV displays in reports and panels. This option has no effect on the way that RUV stores a time value in the repository. Use one of the following values. The default value is **MH:MM:SS**.

- **MH:MM:SS**
  
  Display times in military format, with hours numbered from 00 (midnight) to 23. For example, the third hour past noon is 15:00:00.

- **HH:MM:SS_A/PM**
  
  Display times in AM/PM format, with hours numbered from 1 to 12 (noon or midnight). AM denotes that the time is before noon, and PM denotes that the time is after noon. For example, the third hour past noon is 3:00:00 PM.

**Using the **UPPER_CASE_TRANSATION** Keyword**

Use the **UPPER_CASE_TRANSATION** keyword on the SET DEFAULT or SET command to control whether to display data on reports in mixed case or in uppercase. Case translation processing occurs on all lines directed to the SYSPRINT DD statement. RUV messages that appear in the report are included in the translation process.

Set one of the following values. The default value is **OFF**.

- **OFF** Display data in mixed case.
- **ON** Display in uppercase.
Using the USE_AUTOJOURNAL Keyword

Use the USE_AUTOJOURNAL keyword on the SET DEFAULT or SET command to control whether RUV transforms CICS auto-journaling records into forward-recovery records. This keyword applies to archive log streams only. It has no effect on the processing of RPCV archives, CICS TS journals, or COMPAT41 journals. (See note on page 36 concerning journal support for unsupported CICS releases.)

Set one of the following values for the USE_AUTOJOURNAL keyword. The default value is NO.

NO  Do not transform CICS auto-journaling records into forward-recovery records.

YES  Transform CICS auto-journaling records into forward-recovery records.

Define the file in CICS to match the USE_AUTOJOURNAL specification in RUV. If you specify USE_AUTOJOURNAL(YES), the following minimum CICS file definition options are required for CICS to record before-image records and after-image records:

```
AUTO JOURNALING
J0urnal : 02
JNLRead : Updateonly
JNLSYNCRead : No
JNLUpdate : Yes
JNLAdd : All
JNLSYNCWrite : Yes
RECOVERY PARAMETERS
RECOvery : None
Fwdrecovlog : No
```

The USE_AUTOJOURNAL(YES) keyword is intended for situations in which you cannot afford the CICS overhead of specifying the RECOVERY(ALL) CICS file definition option. The general use of USE_AUTOJOURNAL(YES) is not recommended because data integrity can be compromised.
If these CICS file definition auto-journaling options and the RUV USE_AUTOJOURNAL(YES) keyword are specified during archive creation, RUV produces an archive file with the recoverable data for forward recovery and backout recovery, as shown in the example in Figure 26.

![Figure 26 Sample ARCHIVE SUMMARY Report with USE_AUTOJOURNAL](image)

The BACKOUT RECORDS and FORWARD RECORDS in this sample summary report are the number of transactions that are added, updated, or deleted by CICS and recorded by the CICS Journaling Facility. The existence of the Registered ARCHIVE entry indicates that the archive succeeded. This entry is not displayed if the archive process failed.

In this example, a CICS application performed two add operations, one delete operation, and one update operation for file KSDS01.

Four Adds backout records are created (two before-image records and two after-image records). Three Updates backout records are created (one read-for-update record, one rewrite record, and one read-before-delete record). One Deletes backout record is created. During a backout recovery, two Adds records, one Updates record, and one Deletes record will be backed out.

When USE_AUTOJOURNAL(YES) is specified, RUV builds the after-image records for forward recovery by using the CICS auto-journaling records. In this example, two Adds forward recovery records are created, one Updates record is created, and one Deletes record is created. During a forward recovery, two Adds records and one Updates record will be forward-recovered, and one Deletes record will not be added back to the file.
Using the WORK_DISK Keyword

Use the WORK_DISK keyword to provide RUV with a name (1–8 characters) for temporary disk allocations.

Using the WORK_TAPE Keyword

Use the WORK_TAPE keyword to provide RUV with a name (1–8 characters) for temporary tape allocations.

Using the XBMID Keyword

Use the XBMID keyword on the SET or SET DEFAULT command to provide the subsystem ID of each XBM subsystem that RUV may use for Instant Snapshot copies. You can specify a maximum of 32 XBM subsystem IDs.

List multiple subsystem IDs in the order that you want RUV to attempt to use the identified subsystems. RUV uses the first active XBM subsystem in the list. If no XBM subsystem is active, RUV cannot create Instant Snapshot copies. RUV can restore a data set from an Instant Snapshot copy if no XBM subsystem is available. In this case, RUV performs a VSAM-to-VSAM or sequential-to-sequential file restore rather than an instant restore.

Example of Specifying a Default Set of XBM Subsystems

The following example shows the specification of a default set of XBM subsystems that RUV can use for Instant Snapshot copies. In this example environment, the first subsystem in the list (XBMA) is the production XBM subsystem. The second subsystem (XBMB) is designated to handle Instant Snapshot copy requests if the production subsystem is not active.

```
SET DEFAULT
   XBMID(XBMA XBMB)
```

Example of Specifying a Test XBM Subsystem for the Current Backup Job

The following example shows the specification of a test XBM subsystem (XBMT) for use with the current backup job.

```
SET XBMID(XBMT)
;
```

Viewing Default Settings

You can view default settings by using the REPORT DEFAULT command, as shown in the following example:

```
REPORT DEFAULT;
```

RUV returns a report that shows the current defaults, as shown in the following example:

```
LINE_COUNT ( 79 )
PAGE_WIDTH ( 120 )
DAYS ( 69 )
```

The report displays only the values that have been set explicitly.

Using RUV Default Values

You can clear all values and return to the RUV default values that are provided by using the RESET DEFAULT command, as shown in the following example:

```
RESET DEFAULT;
```

The following message is always returned in response to the RESET DEFAULT command:

```
RUV202094I RUV No default options are set.
```

If you use the REPORT DEFAULT command after you use the RESET DEFAULT command (but before you set any new defaults), RUV issues the same message.
Coding SET DEFAULT Statements

To code a SET DEFAULT statement, use the following syntax.

```
SET DEFAULT
* set default values for RUV options
  { ASSOCIATED_ADDRESS_SPACE_TASKS ( 0 | 10 | nnn | 200 ) }
  { BACKUP_CHANGED_ONLY ( NO | YES ) }
  { BACKUP_METHOD( RUV
    [ BACKUP_REPLY ]
    [ BWO | NO_BWO ]
    [ SNAPSHOT | NO_SNAPSHOT ]
    [ SMS | NO_SMS ]
    [ EXTERNAL | NO_EXTERNAL ]
  ) }
  { DATE_FORMAT ( YYYY.JJ JJJ |
    MM/DD/YYYY |
    DD/MM/YYYY |
    YYYY/MM/DD |
    USER-DEFINED ) }
  { DAYS ( 0 | 45 | nnn | 999 ) | CYCLES ( 0 | nnn | 999 ) }
  { ESOTERIC( device ) }
  { GAP ( STOP | WARN | IGNORE ) }
  { INCREMENTAL_BACKUP_SUPPORT ( YES | NO ) }
  { LANGUAGE ( ENU ) }
  { LINE_COUNT ( 15 | 56 | nnnnnnnn | 99999999 ) }
  { MAX_CC ( nn ) }
  { PAGE_WIDTH ( 30 | nnn | 121 ) }
  { PASSWORD_EXPIRATION_CC ( 0 | 4 ) }
  { <+1> ( 1 | nnnnnnn | 99999999 ) }
  { READER_TASKS ( 1 | nnn | 100 ) }
  { RLS_RECOVER_PROTECTION ( NO | YES ) }
  { SMF_TYPE ( x,y ) }
  { SMF_TYPE x valid values are 0 and 128-255
   SMF_TYPE y valid values are 0 and 1-10
  }
  { SUBSYSTEM ( xxxx ) }
  { SYNTAX_CHECK ( NO | YES ) }
  { TERM_CC ( n1, n2 ) }
  { TERM_CC default value is 4, 0
  }
  { TIME_FORMAT ( HH:MM:SS | HH:MM:SS_A/PM ) }
  { UPPER_CASE_TRANSLATION ( OFF | ON ) }
  { USE_AUTOJOURNAL ( NO | YES ) }
  { WORK_DISK ( devicename8 ) }
  { WORK_TAPE ( devicename8 ) }
  { XBMID ( ssid1 ssid2 ssid3 ... ) }
)
```
To code a SET statement, use the following syntax:

```
SET

* set values for RUV options
  { ASSOCIATED_ADDRESS_SPACE_TASKS ( 0 | 10 | nnn | 200 ) }
  { BACKUP_CHANGED_ONLY ( NO | YES ) }
  { BACKUP_METHOD( 
      RUV
      { BACKUP_REPLY } 
      { BWO | NO_BWO } 
      { SNAPSHOT | NO_SNAPSHOT } 
      { SMS | NO_SMS } 
      { EXTERNAL | NO_EXTERNAL } 
    ) }
  { DATE_FORMAT ( 
      YYYY.JJJ | 
      MM/DD/YYYY | 
      DD/MM/YYYY | 
      YYYY/MM/DD | 
      USER-DEFINED 
    ) }
  { DAYS ( 0 | 45 | nnn | 999 ) | CYCLES ( 0 | nnn | 999 ) }
  { ESOTERIC(device) }
  { GAP ( STOP | WARN | IGNORE ) }
  { INCREMENTAL_BACKUP_SUPPORT ( YES | NO ) }
  { LANGUAGE ( ENU ) }
  { LINE_COUNT ( 15 | 56 | nnnnnnnn | 99999999 ) }
  { MAX_CC ( nn ) }
  { PAGE_WIDTH ( 30 | nnn | 121 ) }
  { PASSWORD_EXPIRATION_CC ( 0 | 4 ) }
  { <+1> ( 1 | nnnnnnnn | 99999999 ) }
  { READER_TASKS ( 1 | nnn | 100 ) }
  { REPORT ( SUMMARY | DETAIL | FULL ) }
  { RLS_RECOVER_PROTECTION ( NO | YES ) }
  { SMF_TYPE ( x,y ) }  *
    SMF_TYPE x valid values are 0 and 128-255
  *
    SMF_TYPE y valid values are 0 and 1-10
  { SUBSYSTEM ( xxxx ) }
  { SYNTAX_CHECK ( NO | YES ) }
  { TERM_CC ( n1, n2 ) }  *
    TERM_CC default value is 4, 0
  { TIME_FORMAT ( MH:MM:SS | HH:MM:SS_A/PM ) }
  { UPPER_CASE_TRANSLATION ( OFF | ON ) }
  { USE_AUTOJOURNAL ( NO | YES ) }
  { WORK_DISK ( devicename8 ) }
  { WORK_TAPE ( devicename8 ) }
  { XBMID ( ssid1 ssid2 ssid3 ... ) }

```
Coding RESET DEFAULT Statements

To code a RESET DEFAULT statement, use the following syntax:

```plaintext
RESET DEFAULT :
```
Working with VSAM File Repository Records

During normal processing, RUV automatically creates VSAM file records in the repository. You can use commands to add VSAM file records before RUV takes action to create the record, to update the information in the record, and to delete the record when it is no longer needed.

Using the ADD VSAM_FILE Command

To add a new VSAM file record to the repository, use the ADD VSAM_FILE command.

This command is used most commonly to handle a data set name in the repository that is listed as --UNKNOWN--. You can add or update the data set name with the correct name, based on the APPLID and FILEID information that is recorded in the repository.

To determine whether the repository contains --UNKNOWN-- data set names, run the REPORT ARCHIVE_FILE(dsn44,*,...) command and use the SET REPORT(DETAIL) option.

The ADD VSAM_FILE command is not related to rule sets, but it allows you to establish purge criteria and APPLID and FILEID relationships.

Coding ADD VSAM_FILE Statements

To code an ADD VSAM_FILE statement, use the following syntax:

```
ADD VSAM_FILE(dsn44)
{ COMMENT('comment fields') }
{ APPLID_FILEID(applid*,fileid*) }
{ DAYS(nn) | CYCLES(nn) }
( SELECTION_EXIT(program8) )
;
```

The following example illustrates how to use the ADD VSAM_FILE command:

```
ADD VSAM_FILE(GNL.LEDGER)
   COMMENT('GENERAL LEDGER FILE')
   APPLID_FILEID(APPLGNL.GNLLEDGR)
   DAYS(999)
;
```
Using the VSAM_FILE Keyword (ADD)

Use the required VSAM_FILE keyword on the ADD VSAM_FILE command to specify the fully qualified data set name of the VSAM file associated with this VSAM file record. Masking (wildcard) characters are not valid on these commands. You can use only one VSAM_FILE keyword and its value on each of these commands.

Use the required VSAM_FILE keyword on the DELETE VSAM_FILE command to specify the data set name of the VSAM file. You can use masking (wildcard) characters in the value. You can use only one VSAM_FILE keyword on this command, but you can code as many values as you need; use a separator between values.

Using the SELECTION_EXIT Keyword (ADD)

You can use the optional SELECTION_EXIT keyword on the ADD VSAM_FILE command to pre-define the name of a user exit routine for future use by RECOVER or PRINT ARCHIVE. See “Using the SELECTION_EXIT User Exit” on page 517. As the value of the SELECTION_EXIT keyword, specify the program name of the exit routine.

Using the APPLID_FILEID Keyword (ADD)

Use the optional APPLID_FILEID keyword on the ADD VSAM_FILE command to identify the CICS APPLID and FILEID associated with the VSAM file.

Specify the APPLID first, then a separator, and then the FILEID (see the example that follows). You can repeat the APPLID_FILEID keyword as many times as required.

```
ADD VSAM_FILE(GNL.LEDGER)
   APPLID_FILEID(APPLID1,FILEID1)
;```

Using Other VSAM_FILE Keywords (ADD)

For more information about using common optional keywords on the ADD VSAM_FILE command, see the following sections:

- “Using the COMMENT Keyword” on page 107
- “Using the CYCLES Keyword” on page 129
- “Using the DAYS Keyword” on page 131

Using the UPDATE VSAM_FILE Command

Update information in a VSAM file record in the repository by using the UPDATE VSAM_FILE command. This command is used most commonly to add or change the DAYS or CYCLES values that control purge processing.

Coding UPDATE VSAM_FILE Statements

To code an UPDATE VSAM_FILE statement, use the following syntax:

```plaintext
UPDATE VSAM_FILE(dsn44)
  { COMMENT('comment fields') } 
  { SELECTION_EXIT(program8) } 
  { APPLID_FILEID(applid,fileid) } 
  { DAYS(nnn) | CYCLES(nnn) } 
 ;
```

The following example illustrates how to use the UPDATE VSAM_FILE command as shown in the following example.

```plaintext
UPDATE VSAM_FILE(GNL.LEDGER) 
  COMMENT('GENERAL LEDGER FILE') 
  SELECTION_EXIT(SELEXIT) 
  APPLID_FILEID(APPLGNL,GNLLEDGR) 
  CYCLES(3) 
 ;
```
Using the VSAM_FILE Keyword (UPDATE)

Use the required VSAM_FILE keyword on the UPDATE VSAM_FILE command to specify the fully qualified data set name of the VSAM file associated with this VSAM file record. Masking (wildcard) characters are not valid on these commands. You can use only one VSAM_FILE keyword and its value on each of these commands.

Using the SELECTION_EXIT Keyword (UPDATE)

You can use the optional SELECTION_EXIT keyword on the UPDATE VSAM_FILE command to change the name of a user exit routine. For more information about the exit routine, see “Using the SELECTION_EXIT User Exit” on page 517. As the value of the SELECTION_EXIT keyword, specify the program name of the exit routine.

Using the APPLID_FILEID Keyword (UPDATE)

Use the optional APPLID_FILEID keyword on the UPDATE VSAM_FILE command to identify the CICS APPLID and FILEID associated with the VSAM file.

Specify the APPLID first, then a separator, and then the FILEID (see the example that follows). You can repeat the APPLID_FILEID keyword as many times as required.

```
UPDATE VSAM_FILE(GNL.LEDGER)
    APPLID_FILEID(APPLID1,FILEID1)
;```

Using Other VSAM_FILE Keywords (UPDATE)

For more information about using common optional keywords on the UPDATE VSAM_FILE commands, see the following sections:

- “Using the COMMENT Keyword” on page 107
- “Using the CYCLES Keyword” on page 129
- “Using the DAYS Keyword” on page 131
Using the DELETE VSAM_FILE Command

To manually remove obsolete or incorrect VSAM file records from the repository, use the DELETE VSAM_FILE command.

Coding DELETE VSAM_FILE Statements

To code a DELETE VSAM_FILE statement, use the following syntax:

```
DELETE VSAM_FILE(dsn44, *, ...)
;  
```

The following example shows how to use the DELETE VSAM_FILE command:

```
DELETE VSAM_FILE(GNL.*);
;  
```

Using the VSAM_FILE Keyword (DELETE)

Use the required VSAM_FILE keyword on the DELETE VSAM_FILE command to specify the data set name of the VSAM file. You can use masking (wildcard) characters in the value. You can use only one VSAM_FILE keyword on this command, but you can code as many values as you need; use a separator between values.
Chapter 4 Using the Batch Journaling Facility

This chapter describes how to set up the RUV Batch Journaling Facility through the use of rule sets.

NOTE
If you have a CICS-only environment (you do not execute batch jobs that update VSAM data sets that you want to protect), you can skip this chapter.

The following information is included:

Overview of Rule Sets ......................................................... 154
Levels of Rule Sets .......................................................... 154
Rule Set Elements ............................................................. 154
Requirements for Rule Set Element Names ......................... 156
Order of Rule Set Entries .................................................. 157
Commands for Use with Rule Set Entries ........................... 158
Enabling Recovery of VSAM Files .............................. 159
Using the ADD VSAM_SET Command ......................... 159
Using the VSAM_RULE Keyword ................................. 159
Coding ADD VSAM_SET Statements .............................. 160
Identifying Jobs, Programs, and Recovery Actions ............. 161
Using the ADD JOB_SET Command ............................. 161
Using the JOB_RULE Keyword ..................................... 161
Using the LOG Keyword .................................................. 162
Using the JOURNAL Keyword ......................................... 162
Using the JOURNAL_ABEND Keyword ......................... 163
Using the JOB_ABEND Keyword ................................. 163
Using the JOB_JCL Keyword .......................................... 163
Using the INTERNAL_READER Keyword ..................... 164
Coding ADD JOB_SET Statements ............................... 164
Constructing Rule Sets ................................................................. 192
Using the ADD RULE_SET Command .............................................. 192
Using the JOB_SET Keyword ....................................................... 192
Using the VSAM_SET Keyword ................................................... 192
Using the USER_VARIABLES Keyword .......................................... 192
Using the EXTERNAL_VENDOR Keyword ....................................... 193
Coding ADD RULE_SET Statements ............................................ 193
Sample Construction of a Rule Set ................................................ 194
Activating a Rule Set ................................................................. 198
Using the ACTIVATE RULE_SET Command .................................. 198
Using the TRACE_ON Keyword .................................................. 199
Processing the ACTIVATE RULE_SET Command .......................... 199
Coding the ACTIVATE RULE_SET Statement ............................... 200
Testing a Rule Set .......................................................................... 201
Coding TRY Statements ................................................................ 201
Using the TRY Command ............................................................ 201
Using the JOB, PROGRAM, and DSN Keywords ......................... 201
Obtaining Detailed Test Results ............................................... 201
Using the ISPF Interface to Work with Rule Sets ......................... 205
Accessing the Subsystem Detail Panel ........................................ 205
Selecting a Type of Rule Set ........................................................ 207
Selecting a Rule Set for Action ..................................................... 208
Creating a New Rule Set ............................................................... 211
Deleting Rule Set Elements .......................................................... 212
Using the DELETE Command ..................................................... 212
Using DELETE Command Keywords ........................................... 212
Using DELETE Command Keyword Values ................................. 213
Coding DELETE Statements ......................................................... 213
Overview of Rule Sets

Rule sets allow you to tailor your batch journaling environment. You tell RUV about the recovery requirements of your VSAM files, and you specify the jobs and programs to which those requirements apply. RUV needs additional information, such as where to store recovery information, how to build backout JCL, and when to run recovery jobs. You use rule sets to provide all of this information.

At installation time, RUV generates rule set samples and forms to help you organize rules for VSAM sets, job sets, and journal models. These samples are located in the RUV.CNTL library.

During the initialization of the BMC Consolidated Subsystem (BCSS), if RUV can not find a rule set, it will generate a subsystem level rule set with the name $DUMMY. The $DUMMY rule set excludes all programs and VSAM files. The $DUMMY rule set will not be registered in the repository.

Levels of Rule Sets

RUV supports two levels of rule sets: a subsystem level and a job level. RUV uses the subsystem level rule set to examine all jobs that are running in the z/OS system. A job level rule set applies to a single batch job only; it does not affect other jobs. RUV searches a job level rule set ahead of a system level rule set. This convention makes testing of rule sets easier and simplifies custom tailoring of rule sets for specific tasks.

Rule Set Elements

Rule sets are a collection of the following types of entries that are stored as records in the RUV repository. You combine the following elements to form a rule set. See Figure 27.

- VSAM rules, which determine recoverability requirements
- job set rules and program rules, which determine when and how logging and journaling occurs by job and by program
- user variables, which provide environmental information
- job JCL, which provides a JCL template for job step backout
- internal readers, which specify what is to be done with a recovery job
- journal models, which provide allocation information for logs and journals
You create rule set entries and combine them into rule sets as described on the following pages:

- VSAM sets on page 159
- job sets on page 161
- journal models on page 166
- user variables on page 171
- job JCL on page 182
- internal readers on page 184
- rule sets on page 192
Requirements for Rule Set Element Names

When you create a rule set element, you assign a name that becomes the name of the element in the repository. The name can be 1 to 32 characters and must be alpha-numeric with no embedded blanks; the underscore (_) character is permitted. You can use the same name for different element types, but duplicate names within the same element type are not permitted.

Do not use EXCLUDE or $DEFAULT as names of rules set elements. EXCLUDE is a reserved action name and cannot be used as a journal model. RUV automatically searches for the following elements if you request that RUV perform step backout for a job and do not provide a specific name:

- LOG_MODEL($DEFAULT)
- INTERNAL_READER($DEFAULT)
- JOB_JCL($DEFAULT)

Labeling Rule Set Names

The labeling scheme you develop for your rule set element names is important. RUV allows you to have duplicate names within a repository provided the names are of different elements. For example, the name of a job set and a VSAM set can be the same since they are different elements. Using the same name allows you to easily identify that these two elements are related. The VSAM set identifies which files are to be controlled and the job set identifies what jobs and programs use those VSAM files.

Reporting Rule Set Names

A labeling scheme can also assist you in rule set reporting. The following labeling scheme process illustrates this concept.

- Start all rules that apply to production files with a P. Start all rules that apply to test files with a T.
- Identify the system by assigning three additional letters to the rule set name: GNL, INV, or PAY.
Order of Rule Set Entries

Further delineate the purpose of this rule set with words added to the name: STANDARD, WEEKLY, or MONTHLY. The following names show the completed label:

- PGNL.STANDARD
- PGNL.MONTHLY
- PINV.DAILY
- TGNL.STANDARD
- TGNL.MONTHLY
- TTST.FRED

Once the naming scheme is complete, you have an easy way to report all production rules by using wildcards:

REPORT VSAM_SET(P*);

Or, if you use a different mask, you can report all production general ledger files:

REPORT VSAM_SET(PGNL*);

Finally, wildcard characters can be placed anywhere in the mask. This makes it simple to find all rules set elements that Fred owns.

REPORT VSAM_SET(*FRED);

Order of Rule Set Entries

The order of the entries in a rule set is important. They dictate the search order that RUV uses to determine rule matching and selection processes. A good practice is to put the EXCLUDE rules before the INCLUDE rules and to put the specific rules before the generic rules.
Commands for Use with Rule Set Entries

To work with rule set entries, you execute the RUV utility program RUVZSM0. You can use the following commands in this utility:

ADD
   Use this command to create a new entry in the repository. This command requires the full entry name.

UPDATE
   Use this command to update an entry in the repository. This command requires the full entry name. In general, you update a rule set in its entirety; you cannot update or replace just a single rule.

DELETE
   Use this command to remove one or more entries from the repository. This command can accept wildcard characters.

REPORT
   Use this command to list information about one or more entries in the repository. This command can accept wildcard characters.

For information about the notation used in the syntax descriptions and the meaning of variable values, see “Syntax Conventions” on page 83.
Enabling Recovery of VSAM Files

To enable RUV recovery of VSAM files, you add one or more VSAM rules to identify VSAM data sets for which you want RUV to record journal and log information. The rule specifies the type of recovery that can be performed against those data sets. You generally group VSAM rules together under one or more VSAM sets.

**NOTE**
See “Daylight Savings Time (DST) Considerations” on page 222 for information about recovery effects caused by resetting the system clock for daylight savings time.

Using the ADD VSAM_SET Command

Use the ADD VSAM_SET command to establish a VSAM set.

Using the VSAM_RULE Keyword

Use one or more VSAM_RULE keywords on the ADD command to identify the VSAM data set and tell RUV what kind of recovery processing you want to enable.

**NOTE**
RUV searches VSAM rules sequentially. Be cautious about the order in which you code them. RUV provides the TRY command as a mechanism for testing rules sets.

Use the following positional values on the VSAM_RULE keyword:

**data set name**
Use the first positional value to identify the VSAM data set name. You can use wildcard characters to specify the data set name. Wildcard characters allow you to identify many VSAM files with fewer rules; they also provide an extra measure of safety because new VSAM data sets will be handled automatically without requiring additional system maintenance.

**recovery type**
Use the second positional value to specify the type of recovery information you want RUV to record to an RUV archive. Use one of the following values. The default value is EXCLUDE.
EXCLUDE
Use this value to prevent journaling of any recovery information. (See “Stopping RUV Activity Immediately”.)

NOTE
Examples of data sets that you do not want to journal are IMS RECON data sets, data sets used by Netview or MQ Series and VSAM data sets used by z/OS subsystems. You will have fewer specifications to code if you use wildcard characters for generic file exclusion.

LOG
Use this value to cause RUV to record before-image records, providing backout capability for the file.

JOURNAL
Use this value to cause RUV to record after-image records providing forward-recovery capability for the file.

BOTH
Use this value to cause RUV to record both before-images and after-images, providing backout and forward-recovery capability for the file.

Coding ADD VSAM_SET Statements

To code an ADD VSAM_SET statement, use the following syntax:

```plaintext
ADD VSAM_SET(name32)
   { COMMENT(comment_fields) }
   VSAM_RULE(dsn_mask44,)
      { EXCLUDE | LOG | JOURNAL | BOTH }
   )
   VSAM_RULE( ... )
;
```

As shown in the following example, use the value of the VSAM_SET keyword to define the name of the VSAM set.

```plaintext
ADD VSAM_SET(PROD_GNL)
   COMMENT("General Ledger System"
            "Accounting department"
            "Production")
   VSAM_RULE(GNLP*.REPORT, EXCLUDE)
   VSAM_RULE(GNLP*, JOURNAL)
;
```
Identifying Jobs, Programs, and Recovery Actions

To identify the jobs and programs that use recoverable VSAM files, and to identify the recovery actions you want RUV to take if an abend occurs, you use job rules. Job rules control when RUV logging and journaling occur by job and by program, as well as where RUV writes the log and journal information. You generally group job rules together under one or more job sets.

The job rule specifies the job and program names. It can provide the name of a template to be used to allocate a log for job step backout and a journal to allocate for forward recovery. It can specify the actions to take if a problem occurs with the journaling process and if a program abend occurs. Job rules can provide other information that is necessary for the job step backout process.

Many programs and utilities should not be logged or journaled. RUV provides a sample exclude job set in member #DEFAULT of the RUV.CNTL library.

Using the ADD JOB_SET Command

Establish a job set by using the ADD JOB_SET command.

Using the JOB_RULE Keyword

Use one or more JOB_RULE keywords on the ADD command. You must provide both of the following values:

job_mask
Use the first positional value to identify the z/OS job name.

program_mask
Use the second positional parameter to identify the z/OS program name.

You can use wildcard characters in these names. Use of wildcards allows you to identify many jobs and programs with fewer rules, as well as provides an extra measure of safety because new jobs and programs will be automatically handled without requiring additional system maintenance.

The following sections describe keywords that you can use with the JOB_RULE keyword.
Using the LOG Keyword

Use the LOG keyword to define the logging environment to set up for a job. A log is a QSAM file that RUV uses to write before-image records of changes to VSAM files. If an abend occurs, the log can be used for job step backout. Use one of the following values for the LOG keyword. The default value is EXCLUDE.

**EXCLUDE**
Use this value to turn off logging. RUV will provide no logging for the job step; step backout will not be performed if an abend occurs.

**journal_model_name**
Use this value to specify the name of a journal model to use as a template for allocating the log. RUV deletes the log file when the step terminates normally. If an abend occurs, RUV catalogs the log file and uses it as input to job step backout.

Use this value in conjunction with the JOB_ABEND keyword, the JOB_JCL keyword, and the INTERNAL_READER keyword to set up the backout environment.

Using the JOURNAL Keyword

Use the JOURNAL keyword to define the journaling environment to set up for a job. Journals are normally used to record after-image records to use for forward-recovery processing; however, they may also contain before-image records. Before-image records on a journal can be used to perform job backout or point-in-time backout over long periods of time. The following values are valid for the JOURNAL keyword. The default value is EXCLUDE.

**EXCLUDE**
Use this value to prevent journal allocation. RUV will provide no journaling for the job step.

**journal_model_name**
Use this value to specify the name of a journal model to use for forward recovery (for saving after-images). The journal model provides a template for RUV to use when allocating the journal.
Using the JOURNAL_ABEND Keyword

Use the JOURNAL_ABEND keyword to tell RUV how to handle a journaling failure. A journaling failure occurs when RUV is unable to allocate a log or journal or cannot record recovery information into a log or journal (usually because of a lack of DASD space or some type of I/O error). The following values are valid for the JOURNAL_ABEND keyword. The default value is ABEND.

ABEND
Use this value to terminate the application program. You should choose to terminate the application program only if RUV performs step backout. RUV backs out all changes, allowing you to correct the journaling problem. You would then rerun the job.

CONTINUE
Use this value to terminate the journaling process and continue with application execution. If RUV terminates the journaling process but the application program continues to run, a data integrity exposure exists until backups of the affected VSAM files are taken.

Using the JOB_ABEND Keyword

If you provide a journal model name for the LOG keyword, use the JOB_ABEND keyword to tell RUV how to handle an application abend. RUV can submit a backout job immediately or can store the backout job JCL in a PDS for later submission. Use one of the following values. The default value is SUBMIT.

SUBMIT
Use this value to submit the backout job immediately. However, RUV does not submit the job if no before-image records were selected to either a log or journal; a WTO will be issued to inform you about the action taken.

NOSUBMIT
Use this value to prevent the submission of a backout job.

Using the JOB_JCL Keyword

If you provide a journal model name for the LOG keyword, use the JOB_JCL keyword to provide the name of the skeleton JCL (a rule set element) that you want to use for building backout JCL and control statements. For information about creating skeleton JCL elements, see “Working with Job JCL Records for Backout” on page 182.
Using the INTERNAL_READER Keyword

If you provide a journal model name for the LOG keyword, use the INTERNAL_READER keyword to specify the name of the internal reader (a rule set element) that you want to use for job step backout processing. The internal reader defines the PDS name and member name to store backout JCL, or it identifies a JES job class for immediate submission of the JCL. For information about internal readers, see “Creating and Updating an Internal Reader” on page 184.

Coding ADD JOB_SET Statements

To code an ADD JOB_SET statement, use the following syntax:

```
ADD JOB_SET(name32)
{ COMMENT(comment_fields) }  
JOB_RULE(job_mask8, program_mask8, 
   { LOG(EXCLUDE | journal_model_name) } 
   { JOURNAL(EXCLUDE | journal_model_name) } 
   { JOB_ABEND(SUBMIT | NOSUBMIT) } 
   { JOB_JCL(name32) } 
   { INTERNAL_READER(name32) } 
   { JOURNAL_ABEND(ABEND | CONTINUE) } 
   ) 
   { JOB_RULE(job_mask8,...) } 
; 
```
As shown in the following example, use the JOB_SET keyword to define the name of the job set.

```
ADD JOB_SET(PROD_GNL)
   COMMENT("Production General Ledger System"
           "Accounting department")
* define end of month jobs
   JOB_RULE(MGNL*    INV987
         LOG(EOM_LOG)
         JOURNAL(EOM_JOURNAL)
         JOURNAL_ABEND(CONTINUE)
         JOB_ABEND(NOSUBMIT)
         JOB_JCL(PROD_BACKOUT)
         INTERNAL_READER(PROD_GNL_PDS)
   )
* define daily jobs
   JOB_RULE(PGNL*    *
         LOG(STANDARD_LOG)
         JOURNAL(LARGE_JOURNAL)
         JOURNAL_ABEND(CONTINUE)
         JOB_ABEND(NOSUBMIT)
         JOB_JCL(PROD_BACKOUT)
         INTERNAL_READER(PROD_GNL_PDS)
   )
```
Defining Journal Models

Use journal models to create templates for constructing a DD allocation statement for QSAM logs and journals.

RUV uses logs to record before-image records, which contain the information that is needed to perform step backout and which are typically discarded after the job step completes successfully. RUV uses journals to record after-image records, which contain the information that is needed for forward recovery, and which are normally used for long-term retention of data.

If the journal data set name and log data set name are the same, RUV creates a blended journal. This journal is retained as a journal, but can be used as a log for backout purposes. Blending logs and journals reduces the amount of I/O and DASD space that otherwise would be required to write the before-image records separately to the log.

The journal model and allocation technique is the same for logs and journals. You will most likely want to use different journal models for logs and journals because of differences in data set naming standards and where the data set is to be allocated. For example, you might want logs to be written to temporary work packs and journals to be written to permanent storage. You might want varying retention periods for each type of data set. The journal data set is always cataloged.

RUV supplies the disposition for log and journal data sets as follows:

- If a job does not abend, the log data set disposition will be altered to DELETE, as in DISP=(,,DELETE).
- If a job abends, RUV ensures that the log data set disposition will be altered to CATLG also, as in DISP=(,,CATLG).

Using the ADD JOURNAL_MODEL Command

Establish a journal model by using the ADD JOURNAL_MODEL command. Use the value of the JOURNAL_MODEL keyword to define the name of the journal model. This name is the same name you use as the value of the LOG or JOURNAL keyword on an ADD JOB_SET statement.

You can use the QSAM keyword to provide allocation information that translates into the fields on a standard DD allocation statement, or you can use the DDNAME keyword to specify the DD name of a statement in the JCL that identifies a data set to use for the output archive file.
You can also use the PROTECTION_LEVEL keyword to balance the level of safety against the level of performance.

The following example shows the use of user variables in the journal model definition with the QSAM keyword. For this example, the user variables <CT>, <PRIMARY>, <SECONDARY>, and <TAPE> must be defined by the active rule set.

```
ADD JOURNAL_MODEL(V11_LOG)
  COMMENT("Tape Journal")
  PROTECTION_LEVEL(0)
  QSAM(
    'DSN=RUVRUV.<JOBNAME>.<+1>.LOG,,' 
    'SPACE=(<CT>,(<PRIMARY>,<SECONDARY>),RLSE),' 
    'DISP=(NEW,DELETE,CATLG),' 
    'UNIT=<TAPE>'
  )

```

The following example shows the use of the DDNAME keyword to override the allocation of the journal or log to use the data set defined with the LOGOUT DD statement:

```
ADD JOURNAL_MODEL(LOGOUT)
  { COMMENT("LOG JOURNAL")  }
  { PROTECTION_LEVEL( 1 ) } 
  DDNAME(LOGOUT)

```

In this example, the following DD statement is defined in the JCL:

```
//LOGOUT DD DSN=RUVRUV.QA.EXT004.JRL,DISP(NEW,CATLG), 
// SPACE=(CYL,(5,5)),UNIT=SYSALLDA,STORCLAS=DEVS90
```

### Using the UPDATE JOURNAL_MODEL Command

Updating an established journal model is similar to adding a journal model, as shown in the following example of the UPDATE JOURNAL_MODEL command. You can specify the PROTECTION_LEVEL keyword and the QSAM keyword or the DDNAME keyword on this command.

```
UPDATE JOURNAL_MODEL(V11_LOG)
  COMMENT("Tape Journal")
  PROTECTION_LEVEL(0)

```
Using the PROTECTION_LEVEL Keyword

Use the optional PROTECTION_LEVEL keyword on the ADD or UPDATE JOURNAL_MODEL command to balance the level of safety against the level of performance. Use one of the following values. The default value is 1.

0  Perform a write-immediate operation (write recovery information to the journal device immediately). This value provides the best safety but results in slower processing because the journal I/O operation must complete before the program can continue. You must use PROTECTION_LEVEL(0) when you are logging or journaling to tape.

1  Perform a write-deferred operation (buffer changes in memory and write them to the journal device when the memory buffer is full). RUV automatically flushes the buffers if the program abends. This value is recommended only if your organization has stable z/OS systems (electrical power loss and unexpected z/OS IPLs do not normally occur). If you are using tape, you must use PROTECTION_LEVEL(0) for data integrity. This specification allows the buffer to write the data to tape if an abend occurs.

2  Reserved for future use.

Using the QSAM Keyword

The QSAM keyword provides allocation information that translates into the fields on a standard DD allocation statement. You provide the data set name and the same space and unit parameters that you would code on a typical z/OS DD statement. The only differences are that the //ddname DD portion is omitted and the data set name and other parameters may contain RUV or IBM substitution variables. Quotes are required where the values contain embedded commas. For more information about variables, see “Creating User Variables” on page 171 and “Using Variables” on page 173.

QSAM allocations may contain any of the standard keywords and parameters listed in Table 6. For more information, see the corresponding JCL topics in the OS/390 documentation set.

**NOTE**

RUV allocates about 200K of memory for buffer storage; consequently, the minimum primary space allocation for a journal should be five tracks (enough to hold 200K of data). A secondary space allocation is acceptable. If the primary allocation is smaller than 5 tracks, the step abends with a D37-04 and user 0116 abend because RUV has determined that not even one file would fit in the allocated space. Use the RLSE parameter to release unused DASD space. Do not code DISP=MOD or use GDGs for logs or journals; RUV allocates a new file for each job step.
Using the DDNAME Keyword

The DDNAME keyword identifies the DD name (1–8 characters) of a DD statement, specified in the execution JCL, that identifies a file for RUV to use as an output archive file. RUV does not dynamically allocate the data set but requires the JCL to identify an appropriate QSAM output file. Use of the DDNAME keyword is mutually exclusive with the QSAM version of the JOURNAL_MODEL command.

**NOTE**

RUV X37 protection is unavailable if you use the DDNAME keyword to override the allocation of journals and logs.

Coding ADD JOURNAL_MODEL Statements

To code an ADD JOURNAL_MODEL statement, use the following syntax:

```plaintext
ADD JOURNAL_MODEL(name32)
   { COMMENT(comment_fields)  }
   { PROTECTION_LEVEL( 1 | 0 ) }
   { QSAM('DSN=dsn44.variables,'  
     'SPACE=(space.variables),'  
     'UNIT=nnnnn,DISP=(nnn)' ) }
   { DDNAME(ddn8)  };
```
To code an UPDATE JOURNAL_MODEL statement, use the following syntax:

```plaintext
UPDATE JOURNAL_MODEL(name32)
  { COMMENT(comment_fields) } 
  { PROTECTION_LEVEL( 1 | 0 ) } 
  { QSAM('DSN=dsn44.variables,' 
       'SPACE=(space.variables),'
       'UNIT=nnnnn,DISP=(nnn)' 
     ) 
  | DDNAME(ddn8) } 
; 
```
Creating User Variables

You can create and use RUV user variables in VSAM name masks in VSAM name models, journal models, internal readers, and job JCL records (skeleton JCL). RUV provides a predetermined set of system variables (see Table 8 on page 174). You can add other user variables as needed. RUV provides a substring capability that allows you to create variables easily.

For information about how to use the user variables you have created, as well as RUV-defined variables, see “Using Variables” on page 173.

Using the ADD_USER_VARIABLES Command

Use the ADD_USER_VARIABLES command to define your user variables in RUV.

Defining User Variables

A user variable consists of a variable name and its variable substitution value. Each user variable is limited to one command line (71 characters).

variable name

The user variable name must begin with the less-than character (<) and end with the greater-than character (>).

variable substitution value

The variable substitution value must follow the variable name and be coded on the same command line. The first non-blank character that follows the greater-than character becomes the delimiter; the next character is the start of the variable substitution value. You must use the same delimiter character at the end of the variable substitution value to designate the end of the value. The variable substitution value itself may not contain any characters that match the selected delimiter character. The variable substitution value may contain other system or user variables.

Using the END_DATA Keyword

Use the END_DATA keyword to indicate the end of the list of user variables.
Working with User Variables

After you have defined user variables, you can work with them in various ways. You can use them as substitution values for journal models, job JCL records, internal readers, and VSAM name masks. You can use substring notation to select a portion of a user variable. You can use IBM-defined variable substitution symbols. You can list the system variables that RUV has defined. For more information about these topics, see “Using Variables” on page 173.

Coding ADD USER_VARIABLES Statements

To code an ADD USER_VARIABLES statement, use the following syntax:

```
ADD USER_VARIABLES(name32)
 { COMMENT(comment_fields) };
<variablename>   /variable1/
<variablename>   /variable2/
END_DATA;
```

As shown in the following example, use the value of the USER_VARIABLES keyword to define the name of the set of user variables.

```
ADD USER_VARIABLES(V11TEST)
 COMMENT("Sample RUV User Variables"
  "Use With RUV $default Rule Sets"
  );
<DASD>   /WORK/
<TR>   /TRK/
<CT>   /CYL/
<PRIMARY_SPACE>   /2/
<SECONDARY_SPACE>   '0'
<ACCOUNT>   '4540'
END_DATA;
```
Using Variables

You can use user-defined variables, RUV-defined variables, and IBM-defined variables when you create journal models, job JCL records (skeleton JCL), internal readers, and VSAM name models. You can use variables to set values for the following elements:

- journal and job names
- dates and times
- user and system identifications
- EXEC statement elements such as program, procedure, and step names

In addition, in backup and restore job JCL records, you can use action variables to control the variable substitution process.

During application program processing, RUV resolves variables to obtain data set names to use in allocation of log, journal, and backup data sets.

You create a user variable by defining the variable name and variable value with the ADD USER_VARIABLES statement, as explained in “Creating User Variables” on page 171.

Including User-Defined and RUV-Defined Variables

To include user-defined or RUV-defined variables in keyword values and JCL, you code the variable name, enclosed by the less-than symbol (<) and the greater-than symbol (>), in the position where you would have coded a literal value, as shown in the following example:

```plaintext
ADD_INTERNAL_READER(GLNRDER)
PDS_OUT(<PROD>.<CNTL>.BKOUTJCL)
MEMBER(<JOB_NUMBER>);
```

You can use substring notation in user-defined and RUV variables, as explained in “Using Substring Notation” on page 174.
Using Substring Notation

You can select portions of a user variable by using substring notation, as shown in the following example:

\[ \text{<USER_VARIABLE \{ (start_location \{, length \} \) \}} > \]

The `start_location` and `length` values indicate position and count. The default `start_location` number is 1. If the `start_location` number is larger than the length of the variable, RUV substitutes a null value. Use a minus sign in the `start_location` value to specify a count from the end of the variable and select characters to the left by the `length` value. If you do not provide a `length` value, RUV assumes that you want to use the remainder of the string.

Table 7 shows examples using a job name of PRODGNL.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Result</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;PRODGNL&gt;</code></td>
<td>PRODGNL</td>
<td>default uses all characters</td>
</tr>
<tr>
<td><code>&lt;PRODGNL(3,2)&gt;</code></td>
<td>OD</td>
<td>start at third character, count 2</td>
</tr>
<tr>
<td><code>&lt;PRODGNL(,2)&gt;</code></td>
<td>PR</td>
<td>start at first character, count 2</td>
</tr>
<tr>
<td><code>&lt;PRODGNL(5)&gt;</code></td>
<td>GNL</td>
<td>start at fifth character</td>
</tr>
<tr>
<td><code>&lt;PRODGNL(9,3)&gt;</code></td>
<td></td>
<td>9 too large, null substitution</td>
</tr>
<tr>
<td><code>&lt;PRODGNL(-3)&gt;</code></td>
<td>GNL</td>
<td>go to end, move back 3</td>
</tr>
</tbody>
</table>

Using RUV-Defined Variables

Table 8 lists all RUV-defined variables. You use them in the same way as you use user-defined variables. RUV also provides action variables, in the format `<:variable>`, which are listed in “Using Action Variables” on page 179.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;BACKUP_DSN&gt;</code></td>
<td>The data set name of the backup data set. This variable is the value of the BACKUP_FILE keyword (1–44 characters) from the REGISTER BACKUP_FILE command that was used to register the backup. This variable is valid in restore job JCL records.</td>
</tr>
<tr>
<td><code>&lt;BACKUP_TIME&gt;</code></td>
<td>The start time, in <code>yyyyjjjhhmmsst</code> format, of the backup. This variable is the value of the START_TIME keyword. This variable is valid in restore job JCL records.</td>
</tr>
<tr>
<td><code>&lt;JOBNAME&gt;</code></td>
<td>Standard z/OS JCL job statement name that executes an initiator.</td>
</tr>
</tbody>
</table>
### Table 8  RUV-Defined Variables (Part 2 of 3)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;JOB_NUMBER&gt;</td>
<td>Standard z/OS JES job number.</td>
</tr>
<tr>
<td>&lt;LOOP_TIME&gt;</td>
<td>A time, in yyyyjjjhhmmsst format, that RUV assigns to this variable when it encounters the &lt;BEGIN_LOOP&gt; action variable. This time value does not change until RUV finishes processing the members of the group and exits the loop.</td>
</tr>
<tr>
<td>&lt;LOOP_TIME_HEX&gt;</td>
<td>A time, in Xhhhhhhhhhhhhhhhhhh format, that RUV assigns to this variable when it encounters the &lt;BEGIN_LOOP&gt; action variable. This time value does not change until RUV finishes processing the members of the group and exits the loop.</td>
</tr>
<tr>
<td>&lt;NODEn&gt;</td>
<td>The contents of the node (qualifier) in the input data set name at the position that is indicated by n; for example, if the input data set name is PAYROLL.TAXWTHLD.DATA, the variable &lt;NODE2&gt; refers to the string TAXWTHLD. The &lt;NODEn&gt; variable is valid in VSAM name models, output models, and job JCL records. If you specify this variable in a backup or restore job JCL record, it is valid only within a set of &lt;BEGIN_LOOP&gt; and &lt;END_LOOP&gt; action variables.</td>
</tr>
<tr>
<td>&lt;PARAMETER&gt;</td>
<td>A 64-byte user-defined field. The value is provided at registration time. This variable is valid in backup and restore job JCL records.</td>
</tr>
<tr>
<td>&lt;PGMNAME&gt;</td>
<td>Standard z/OS JCL EXEC job statement name for program being executed.</td>
</tr>
<tr>
<td>&lt;PROCNAME&gt;</td>
<td>Standard z/OS JCL procedure name in a PROCLIB that will produce a set of JCL.</td>
</tr>
<tr>
<td>&lt;RUVBACKUP&gt;</td>
<td>consolidates &lt;RUVDATE&gt; and &lt;RUVTIME&gt; into one variable.</td>
</tr>
<tr>
<td>&lt;RUV_CONTROL&gt;</td>
<td>A place-holder specification for all RUV commands and keywords. Used in JCL and ISPF. This variable indicates the position in which all control statements will be placed in your job.</td>
</tr>
<tr>
<td>&lt;RUVDATE&gt;</td>
<td>The current Julian date Dyyyyjjj: 4-digit year, 3-digit day of the year (1 to 366).</td>
</tr>
<tr>
<td>&lt;RUV_JOB_JCL&gt;</td>
<td>The 32-character name of the backup job JCL record that was specified in the SUBMIT command.</td>
</tr>
<tr>
<td>&lt;RUV_JOURNAL&gt;</td>
<td>The journal data set name (44-characters) that is active for the batch journaling facility. Whenever a batch job is running and it has journaling, a user-set file will be opened named RUV_JOURNAL. Used primarily for backout situations.</td>
</tr>
</tbody>
</table>
| <RUV_JOURNAL_DEACTIVATE> | Used to deactivate logs and journals after a backout. Used in JOB_JCL models; valid only in backout jobs. Causes the following statement to be added for each log or journal created in a job step:  
  UPDATE ARCHIVE_FILE(archive_dsn)  
  REGISTRATION_TIME(Xhhhhhhhhhhhhhhhhhh) STATUS(INACTIVE);  
  Xhhhhhhhhhhhhhhhhhh is replaced with the registration time of the archive or archives. |
## Table 8  RUV-Defined Variables (Part 3 of 3)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;RUV_LOG&gt;</td>
<td>The logging data set name (44-characters) that is active for the batch journaling facility. Whenever a batch job is running and it has before-image logging, a user-set file will be opened named RUV_LOG. Used primarily for backout situations.</td>
</tr>
<tr>
<td>&lt;RUV_STEPLIB&gt;</td>
<td>Propagates the current z/OS load libraries and RUVEVLIB to the recovery job. Used in JOB_JCL models. Enter this variable on a line by itself. It might produce multiple lines of output.</td>
</tr>
<tr>
<td>&lt;RUVTIME&gt;</td>
<td>The current GMT time in 24-hour military time: Thhmmssst 2-digit hour (00 to 23), 2-digit minute (00 to 59), 2-digit second (00 to 59), and 1-digit tenths of a second (0 to 9).</td>
</tr>
<tr>
<td>&lt;RUVTIME_HEX&gt;</td>
<td>The current GMT time in hexadecimal format: Xhhhhhhhhhhhhhhhhh.</td>
</tr>
<tr>
<td>&lt;SMFID&gt;</td>
<td>Standard z/OS JCL 4-character name that identifies a system (like SYSx).</td>
</tr>
<tr>
<td>&lt;STEPNAME&gt;</td>
<td>Standard z/OS JCL job step name that executes in an initiator.</td>
</tr>
<tr>
<td>&lt;SYSNDX&gt;</td>
<td>A seven-digit number that works with &lt;:BEGIN_LOOP&gt; and &lt;:END_LOOP&gt; action variables that define a processing loop. &lt;BEGIN_LOOP&gt; sets the value of &lt;SYSNDX&gt; to 0000001. RUV processes the statements within the loop for each member of a VSAM group and increments the value of &lt;SYSNDX&gt; by one for each loop.</td>
</tr>
<tr>
<td>&lt;SYSNDX&gt; is valid in backup and restore job JCL records.</td>
<td></td>
</tr>
<tr>
<td>&lt;USERID&gt;</td>
<td>Standard z/OS JCL job statement value that identifies the user.</td>
</tr>
<tr>
<td>&lt;VSAM_DEFINITION&gt;</td>
<td>Build IDCAMS Define function statements for a VSAM file. This variable is valid only within a set of &lt;BEGIN_LOOP&gt; and &lt;:END_LOOP&gt; action variables.</td>
</tr>
<tr>
<td>&lt;VSAM_PATH_DEFINITION&gt;</td>
<td>Build IDCAMS Define function statements for all associated alternate indexes and paths. BLDINDEX commands are also created. This variable is valid only within a set of &lt;BEGIN_LOOP&gt; and &lt;:END_LOOP&gt; action variables.</td>
</tr>
<tr>
<td>&lt;VSAM_DSN&gt;</td>
<td>A user-specified data set name (44-characters). If you specify this variable in a backup or restore job JCL record, it is valid only within a set of &lt;BEGIN_LOOP&gt; and &lt;:END_LOOP&gt; action variables.</td>
</tr>
<tr>
<td>&lt;+1&gt;</td>
<td>This variable allows you to change a value incrementally by adding a number to the end of it. Each time you use this variable, it adds 1 to a hidden number n (a 7-character number formatted as V0000000). The first time the variable is used, it is incremented to 1 (V0000001). The next time it is used, it is 2 (V0000002), and so forth.</td>
</tr>
</tbody>
</table>
Using the <+1> Variable for X37 Support

Each time the <+1> variable is used in variable substitution, the associated number is incremented by one. The <+1> variable is resolved to a V followed by seven digits. “GNLP.<RUVDATE>.<+1>” would become “GNLP.D1998273.V0000001”, then “GNLP.D1998273.V0000002” and so on. The number is incremented for each substitution reference; consequently, “PAY.<+1>.<+1>” would become “PAY.V0000003.V0000004”. Substring notation is permitted.

RUV uses the presence of the <+1> variable in a log or journal DSN to determine whether you want X37 protection. When the <+1> variable is used, RUV automatically allocates additional logs or journals if a log or journal fills and an X37 abend is detected.

When the <+1> variable is used to generate a journal or log DSN, the z/OS catalog is checked for a duplicate DSN. If duplicate names would occur, the number is incremented and the catalog check performed until the duplicate name condition does not occur.

Changing the Value of the <+1> Variable

The <+1> variable is stored on the RUV repository. A repository restore will cause the value of the <+1> variable to be reset to the value it had at the time of the repository backup. You may change the value of the <+1> variable by the following control statement:

```
SET <+1>(nn);
```

The next usage of the <+1> variable will generate a value of $nn + 1$.

The $nn$ value may be any value from 1 to 9,999,999.

Record Count Discrepancies

The real record counts and the actual records that RUV writes to a journal or log may be higher than expected. This condition may occur when the output gets an X37-type abend. RUV automatically writes all the records from storage buffers to the new file. This action causes duplicate records in the output, and therefore increased record counts when RUV reads these files during the recovery process. The duplicate records cause no problem to the recovery process but cause the record count to be incorrect for the file that receives the X37-type abend. Normal processing does not cause record counts to increase.
Allocation of Batch Logs

It is strongly recommended that you use the <+1> variable when allocating batch logs. RUV must be able to record backout information on the batch log before a successful backout can be run. If all backout information cannot be recorded (the log had a media failure or an X37 abend), the log data will be written to the repository.

Using IBM Variable Substitution Symbols

RUV supports IBM-type variable (such as &XXX.&YYY) substitution, as shown in the following example:

GL.&SYSUID.&DATE.T<RUVTIME(2,5)>

You can display the IBM symbols that are defined at your data center by entering the command D SYMBOLS at the z/OS console. The standard IBM variables are documented in the z/OS MVS Initialization and Tuning Reference manual in the topic “Dynamic System Symbols.”

Listing RUV System Variables

RUV provides preset system user variables. You can obtain a list of RUV system variables by entering the REPORT SYSTEM_VARIABLES command. The command produces the following output in the SYSPRINT data set. The <RUV_JOURNAL> and <RUV_LOG> definitions will be filled in correctly when a recovery job is being built.

Figure 28  REPORT SYSTEM_VARIABLES Command Output

<table>
<thead>
<tr>
<th>SYSTEM_VARIABLES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;RUVDATET</td>
<td>/D2009246/</td>
</tr>
<tr>
<td>&amp;RUVTIMET</td>
<td>/T1753342/</td>
</tr>
<tr>
<td>&amp;SMFID</td>
<td>/IMSA/</td>
</tr>
<tr>
<td>&amp;SSID</td>
<td>/VRM1/</td>
</tr>
<tr>
<td>&amp;JOBNAME</td>
<td>/ISIRAMRP/</td>
</tr>
<tr>
<td>&amp;PROCNAME</td>
<td>/REPORT/</td>
</tr>
<tr>
<td>&amp;STEPNAME</td>
<td>/REPORT/</td>
</tr>
<tr>
<td>&amp;PGMNAME</td>
<td>/RUVZSM0/</td>
</tr>
<tr>
<td>&amp;USERID</td>
<td>/ISIRAM3/</td>
</tr>
<tr>
<td>&amp;JOB_NUMBER</td>
<td>/JOB08406/</td>
</tr>
<tr>
<td>&amp;RUV_JOURNAL</td>
<td>/NULLFILE/</td>
</tr>
<tr>
<td>&amp;RUV_LOG</td>
<td>/NULLFILE/</td>
</tr>
<tr>
<td>&amp;RUV_CONTROL</td>
<td>/ /</td>
</tr>
<tr>
<td>&amp;+1</td>
<td>/V0000000/</td>
</tr>
</tbody>
</table>
Using Action Variables

You can use RUV action variables in backup and restore job JCL records to control the variable substitution process.

Action Variable Syntax

Action variables are enclosed by the less-than symbol and the colon (<:) and a greater-than (>) symbol; <BEGIN_LOOP> is an example of an action variable.

NOTE

The < combination is reserved by RUV and is always interpreted as an action variable. Do not use this combination for any other purpose.

RUV Action Variables

Table 9 lists the RUV action variables that you can use.

Table 9  RUV Action Variables (Part 1 of 2)

<table>
<thead>
<tr>
<th>Action Variable</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BEGIN_LOOP&gt;</td>
<td>Code the &lt;BEGIN_LOOP&gt; action variable at the beginning of a set of JCL statements that you want RUV to process for each member of the specified VSAM group. &lt;BEGIN_LOOP&gt; also resets the &lt;SYSNDX&gt; variable to 0000001.</td>
</tr>
<tr>
<td></td>
<td>Loops cannot be nested; close a loop that is started with the &lt;BEGIN_LOOP&gt; action variable by coding the &lt;END_LOOP&gt; action variable before you code another &lt;BEGIN_LOOP&gt; action variable.</td>
</tr>
<tr>
<td>&lt;END_LOOP&gt;</td>
<td>Code the &lt;END_LOOP&gt; action variable at the end of a set of JCL statements that you want RUV to process for each member of the specified VSAM group.</td>
</tr>
<tr>
<td>&lt;IMBED imbed_source&gt;</td>
<td>Code the &lt;IMBED outside_source&gt; action variable to insert statements from a source that is outside of the job JCL record. The string outside_source represents the ddname or data set name of a flat file or partitioned data set member that contains fixed-length, 80-byte records. After RUV inserts the statements from the outside source, RUV continues to perform variable substitution for any variables. RUV supports nested &lt;IMBED outside_source&gt; action variables for a maximum of ten levels. You can use variables to build the ddname or data set name that identifies the outside source. Code the &lt;IMBED outside_source&gt; action variable on a line by itself. You can also use the &lt;IMBED outside_source&gt; action variable with the STORE INTO_FILE command.</td>
</tr>
</tbody>
</table>
Examples of Action Variable Usage

The following examples demonstrate the use of RUV action variables in job JCL records.

::<VSOFF> and ::=VSON

In the following example, the ::=VSOFF> and ::=VSON> action variables are used in a comment to document how a data set name was constructed.

```plaintext
...  
//DDNAME1 DD DISP=SHR,DSN=<NODE1>.<NODE2>.BACKUP
...  
...  
//DDNAME1 DD DISP=SHR,DSN=APY.PAYROLL.BACKUP
...  
...  
//DLETCMDS DD DSN=IDCAMS.DELETE.COMMANDS(APYDLET),DISP=SHR
...  
::<IMBED DLETCMDS>
...  
```

If the input data set name is APY.PAYROLL.EMPL, the following statements result when RUV performs variable substitution:

```plaintext
...  
//DDNAME1 DD DISP=SHR,DSN=<NODE1>.<NODE2>.BACKUP
...  
...  
//DDNAME1 DD DISP=SHR,DSN=APY.PAYROLL.BACKUP
...  
...  
//DLETCMDS DD DSN=IDCAMS.DELETE.COMMANDS(APYDLET),DISP=SHR
...  
::<IMBED outside_source>
...  
```

In the following example, the ::=IMBED outside_source> variable is used to insert a set of IDCAMS Delete statements from a member of a partitioned data set into the generated job.

```plaintext
...  
//DLETCMDS DD DSN=IDCAMS.DELETE.COMMANDS(APYDLET),DISP=SHR
...  
::<IMBED DLETCMDS>
...  
```
<:BEGIN_LOOP> and <:END_LOOP>

The example in “Creation of a Job JCL Record to Perform Backups” on page 319 shows the use of the <:BEGIN_LOOP> and <:END_LOOP> action variables.
Working with Job JCL Records for Backout

The following sections explain how to work with job JCL records, also known as skeleton JCL, to execute recovery for job step backout.

RUV also uses job JCL records to contain skeleton JCL for executing VSAM file backups and restores that are performed by processes that are external to RUV (for external backup support). For more information, see “Working with Job JCL Records for Backup and Restore Processes” on page 327.

Using the ADD JOB_JCL Command

Use the ADD JOB_JCL command to create skeleton JCL, as shown in the following example. Use the JOB_JCL keyword to define the name of the job JCL rule. (This name is the same name you use as the value of the JOB_JCL keyword on an ADD JOB_SET statement.)

NOTE
There is a limit of 400 statements per job.

```
//SYSIN   DD DATA,DLM= $$
ADD JOB_JCL(PROD_BACKOUT):
  //<JOBNAME(1,8)> JOB (<ACCOUNT>),CLASS=A,
  //   MSGCLASS=X,
  //   REGION=3M
  //* PRERECOVERY STEPS AS REQUIRED BY USER
  //ZSM0  EXEC PGM=RUVZSMO
  <RUV_STEPLIB>
  //SYSPRINT DD SYSOUT=* 
  //SYSIN   DD *
  <RUV_CONTROL>
  <RUV_JOURNAL_DEACTIVATE>
  /*
  //* POST RECOVERY STEPS AS REQUIRED BY USER
END_DATA;
  $$
```

The ADD JOB_JCL stores the skeleton JCL in the repository. Use one of the following methods to place the JOB_JCL into the repository:

- Store the recovery JCL in a PDS, and point SYSIN to the PDS member.
Place the recovery JCL in-stream with other RUV commands, and use the //SYSIN DD DATA,DLM=\$\$ statement to allow JES to recognize the recovery JCL as input data.

**Defining Skeleton JCL Statements**

Skeleton JCL statements are similar to the standard statements you would code for a backout job, except that you can use RUV variables.

At execution, RUV replaces the <RUV_CONTROL> variable with backout control statements that are generated by RUV for the failing program. RUV can submit the generated JCL directly to JES or store the JCL in a PDS. This action is determined by the INTERNAL_READER keyword value of the JOB_RULE statement.

You can include sort parameters in the skeleton JCL. For more information, see “Sort Parameters” on page 472.

In a backout job, you can use the following RUV-defined variables. For more information, see Table 8 on page 174.

- <RUV_JOURNAL_DEACTIVATE>
- <RUV_STEPLIB>

**Using the END_DATA Keyword**

Use the END_DATA keyword to indicate the end of the skeleton JCL.

**Coding ADD JOB_JCL Statements**

To code an ADD JOB_JCL statement, use the following syntax:

```plaintext
ADD JOB_JCL(name32)
   | COMMENT(comment_fields) );
//<JOBNAME(1,8)> JOB (<ACCT>),CLASS=A, ...
//RUVZSMO EXEC PGM=RUVZSMO
<RUV_STEPLIB>
//SYSPRINT DD SYSOUT=*  
//SYSIN   DD *  
<RUV_CONTROL>
<RUV_JOURNAL_DEACTIVATE>
END_DATA;
```
Creating and Updating an Internal Reader

Use internal reader rules to identify where to send a recovery job in the event of an abend. You can have RUV write recovery JCL to a PDS member, write it to a data set, or submit it to JES.

Using the ADD INTERNAL_READER Command

Establish an internal reader by using the ADD INTERNAL_READER command, as shown in the following example. (The following example shows the use of user variables in the internal reader definition.)

```
ADD INTERNAL_READER(READER1)
  CLASS(A)
  COMMENT(Reader 1)
;
ADD INTERNAL_READER(READER2)
  PDS_OUT(<HIGHQUAL>.<PDS2>)
  MEMBER(<JOB_NUMBER>)
  COMMENT(Reader 2)
;
ADD INTERNAL_READER(READER3)
  DDNAME(RDR3)
  COMMENT(Reader 3)
;
```

Use the value of the INTERNAL_READER keyword to define the name of the internal reader. This name is the same name you use as the value of the INTERNAL_READER keyword on an ADD JOB_SET statement.

You can code either the CLASS keyword, the PDS_OUT and MEMBER keywords, or the DDNAME keyword on each ADD INTERNAL_READER command.
Using the UPDATE INTERNAL_READER Command

Updating an established internal reader is similar to adding an internal reader, as shown in the following example:

```plaintext
UPDATE INTERNAL_READER(READER1)
  CLASS(A)
  COMMENT('Reader 1')
;
UPDATE INTERNAL_READER(READER2)
  PDS_OUT(<HIGHQUAL>.<PDS2>)
  MEMBER(<JOB_NUMBER>)
  COMMENT('Reader 2')
;
UPDATE INTERNAL_READER(READER3)
  DDNAME(RDR3)
  COMMENT('Reader 3')
;
```

Using the CLASS Keyword

Use the CLASS keyword to provide the JES job class (z/OS internal reader) to use for immediate submission of the recovery job.

Using the PDS_OUT and MEMBER Keywords

Use the PDS_OUT keyword to provide the name of an existing data set to contain the generated recovery job. Use the MEMBER keyword to provide the member name you want to use for the generated job; if the member already exists when RUV generates a new recovery job, RUV overwrites the existing member. You can include variables in the data set name and member name, as explained in “Using Variables” on page 173.

The PDS_OUT data set have a fixed blocked record format (RECFM=FB) and a logical record length of 80 (LRECL=80). Other formats are not supported. The PDS_OUT data set cannot be a PDS/E.

Using the DDNAME Keyword

Use the DDNAME keyword to provide the DD name (1–8 characters) of an output data set to contain the generated recovery job. RUV does not dynamically allocate the data set but requires the JCL to identify an appropriate QSAM output file.
Coding ADD INTERNAL_READER Statements

To code an ADD INTERNAL_READER statement, use the following syntax:

```
ADD INTERNAL_READER(name32)
   { COMMENT(comment_fields) }  
   { PDS_OUT(pds.dsn44.with.variables)  
     MEMBER(var_permitted) }  
   | { CLASS(a) }  
   | { DDNAME(ddn8) } 
;  
```

Coding UPDATE INTERNAL_READER Statements

To code an UPDATE INTERNAL_READER statement, use the following syntax:

```
UPDATE INTERNAL_READER(name32)
   { COMMENT(comment_fields) }  
   { PDS_OUT(pds.dsn44.with.variables)  
     MEMBER(var_permitted) }  
   | { CLASS(a) }  
   | { DDNAME(ddn8) } 
;  
```

As shown in the following example, you can use the INTERNAL_READER keyword to specify the name and location of the PDS member.

```
ADD INTERNAL_READER( PROD_RECOVER_PDS )
   PDS_OUT( PROD_RECOVERY.JCL )  
   MEMBER( <JOB_NUMBER> ) 
;  
```
Creating External Vendors

RUV allows an interface to products created by other vendors. Functional events in RUV such as ADD, UPDATE, DELETE, journaling, and registration activities can be provided to an external vendor exit program as data records.

**NOTE**

RUV activates external vendor exits from APF authorized libraries only. In the job step JCL, an RUVEVLIB DD statement may be specified. At rule set activation, all libraries in the DD statement must be verified as APF authorized. All exit programs will be loaded from that DD statement. Data records provided to an external vendor are available as read only information. The RUV-provided exit, RUVSTAT, does not require the RUVEVLIB DD statement.

When z/OS initializes, all standard subsystems and address spaces are started. BCSS locates all external vendor records within a rule set and reads the repository to determine whether certain events are to be enabled.

From those external vendor records, BCSS builds an external vendor list with an entry for each of the specified vendors. Only active vendors are used. The external vendor interface is then ready to process external vendor programs.

**Using the ADD EXTERNAL_VENDOR Command**

Use the ADD EXTERNAL_VENDOR command to create an external vendor program record. This record will be used by the ACTIVATE RULE_SET command if any external vendors are identified in the rule set being activated.

To code an ADD EXTERNAL_VENDOR statement, use the following syntax:

```
ADD EXTERNAL_VENDOR(name32)
{ COMMENT(comment_fields) }
{ STATUS(ACTIVE | INACTIVE) }
{ STATISTICS(NO | YES) }
{ BACKUP_REGISTRATION(NO | program8, “initialization parameters”) }
{ JOURNAL_REGISTRATION(NO | program8, “initialization parameters”) }
;
```
Using the UPDATE EXTERNAL_VENDOR Command

Use the UPDATE EXTERNAL_VENDOR command to update an external vendor program record. This command uses the same keywords and values that the ADD EXTERNAL_VENDOR command uses.

```
UPDATE EXTERNAL_VENDOR(name32, * , ...) 
   { COMMENT(comment_fields) } 
   { STATUS(ACTIVE | INACTIVE) } 
   { BACKUP_REGISTRATION(NO | program8 
     { "initialization parameters" } ) ) } 
   { JOURNAL_REGISTRATION(NO | program8) } 
   { "initialization parameters" } ) ) }
```

Using the STATUS Keyword

The STATUS keyword activates or deactivates use of the EXTERNAL_VENDOR.

The following values are valid for the STATUS keyword. The default value is ACTIVE.

**ACTIVE**
Use this value to activate use of the EXTERNAL_VENDOR.

**INACTIVE**
Use this value to deactivate use of the EXTERNAL_VENDOR.

Using the BACKUP_REGISTRATION Keyword

The BACKUP_REGISTRATION keyword specifies external vendor exit backup programs and parameters. Two or more EXTERNAL_VENDOR commands can have backup registration keywords that refer to the same program.

The following values are valid for the BACKUP_REGISTRATION keyword. The default value is NO.

**NO**
Use of this value prevents EXTERNAL_VENDOR from notifying an external vendor exit that RUV backup data is available.
Using the JOURNAL_REGISTRATION Keyword

The JOURNAL_REGISTRATION keyword specifies user exit journal programs and parameters. Two or more EXTERNAL_VENDOR commands can have journal registration keywords that refer to the same program.

The following values are valid for the JOURNAL_REGISTRATION keyword. The default value is NO.

**NO** Use of this value prevents EXTERNAL_VENDOR from notifying an external vendor exit that RUV journal data is available.

**program8** Use of this value allows EXTERNAL_VENDOR to notify an external vendor exit that journal data is available.

**Initialization Parameters**
Actions of the external vendor’s program customize the initialization parameters. The external vendor is responsible for providing you with this information.

Using the RUV-Provided External Vendor

RUV provides an exit program called RUVSTAT that can be used for the program8 value. The EXTERNAL_VENDOR program RUVSTAT will provide batch job statistics at step termination. Under optional initialization parameter control, you specify that RUVSTAT print this journal information to the CONSOLE or JESYSMSG.

The following example specifies the following items:

- The default external vendor (RUVSTAT) is active.
- Statistics will be reset to 0 and gathered.
- Registration of journal statistics is printed to JESYSMSG.
**Figure 29 Sample RUVSTAT Printout**

<table>
<thead>
<tr>
<th><strong>RECOVERY UTILITY FOR VSAM STATISTICS</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JOBNAME</strong>: DOPRNT1 <strong>STEPNAME</strong>: PRTARCH</td>
<td></td>
</tr>
<tr>
<td><strong>SUBSYSTEM</strong>: RUV1</td>
<td></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUVEBASE</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>0 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUVESMS0</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>0 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUVE0010</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>0 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUVKBASE</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>1 Adds</strong></td>
<td><strong>0 Updates</strong></td>
</tr>
<tr>
<td><strong>6 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUKDPV0</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>6 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUKPTH1</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>1 Adds</strong></td>
<td><strong>1 Updates</strong></td>
</tr>
<tr>
<td><strong>0 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUKRKP9</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>6 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUUKSMS0</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>6 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUK0010</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>6 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RURUVMS0</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>6 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
<tr>
<td><strong>DSN</strong>: RUVRUV.DV701VP.RUR0010</td>
<td></td>
</tr>
<tr>
<td><strong>1 Opens</strong></td>
<td><strong>1 Closes</strong></td>
</tr>
<tr>
<td><strong>4 Adds</strong></td>
<td><strong>2 Updates</strong></td>
</tr>
<tr>
<td><strong>6 Deletes</strong></td>
<td><strong>0 Discards</strong></td>
</tr>
</tbody>
</table>
Activating the EXTERNAL_VENDOR

The ACTIVATE RULE_SET command activates an ADD EXTERNAL_VENDOR command or an UPDATE EXTERNAL_VENDOR command.

**NOTE**
If the *name32* field contains the keyword $LAST_VENDOR, no additional scanning for EXTERNAL_VENDORs occurs past that record. This is not useful for system level rule sets. $LAST_VENDOR should not be used to specify an external vendor.

Using the DELETE EXTERNAL_VENDOR Command

Use the DELETE EXTERNAL_VENDOR command to remove an interface with an external vendor program. See “Deleting Rule Set Elements” on page 212 for coding instructions.
Constructing Rule Sets

A rule set defines the VSAM data sets to be protected and environments those data sets will be selected from and saved in. To create a rule set, you construct, at a minimum, a VSAM set and job set.

Rule sets that consist of elements other than those minimum elements can be used as job-level rule sets. You can use partial job-level rule sets to override the system-level rules in effect. For example, with a partial job-level rule set, you can enlarge the size of journals for special processing needs.

When you construct and code VSAM sets and job sets, and also include user variables, journal models, skeleton JCL, internal readers, and external vendors, you have produced all of the rule set pieces available for constructing rule sets.

Using the ADD RULE_SET Command

Use the ADD RULE_SET command to create and install a rule set to the repository. The installed rule set can be selected and activated when it is needed.

Using the JOB_SET Keyword

Use the required JOB_SET keyword to provide a list of the job sets that you want to use in this job rule. (The JOB set name is the value of the JOB_SET keyword on the ADD JOB_SET statement.)

Using the VSAM_SET Keyword

Use the required VSAM_SET keyword to provide a list of the VSAM sets that you want to use in this job rule. The value is one or more VSAM set names. (The VSAM set name is the value of the VSAM_SET keyword on the ADD VSAM_SET statement.)

Using the USER_VARIABLES Keyword

Use the optional USER_VARIABLES keyword to provide a list of the user variable sets that you want to use in this job rule. The value is one or more user variable set names. (The user variable set name is the value of the USER_VARIABLES keyword on the USER_VARIABLES statement.)
Using the EXTERNAL_VENDOR Keyword

Use the EXTERNAL_VENDOR keyword to provide a list of external vendor interfaces. See “Creating External Vendors” on page 187.

Coding ADD RULE_SET Statements

To code an ADD RULE_SET statement, use the following syntax:

```
ADD RULE_SET(name32)
  { JOB_SET(name32, ...) }
  { VSAM_SET(name32, ...) }
  { COMMENT(comment_fields) }
  { USER_VARIABLES(name32, ...) }
  { EXTERNAL_VENDOR(name32, ...) }
;
```

Use the value of the RULE_SET keyword to define the name of the rule set (see the example that follows). This name is the name that you use as the value of the RULE_SET keyword on an ACTIVATE RULE_SET statement."

```
ADD RULE_SET(TEST_RULES)
    COMMENT("Test GNL rules")
    JOB_SET(PROD_GNL
        $DEFAULT_EXCLUDES)
    VSAM_SET(PROD_GNL
        $EXCLUDES)
    USER_VARIABLES(PROD_VARIABLES)
    EXTERNAL_VENDOR($DEFAULT)
;
```

**NOTE**

The order of the JOB_SET values and the VSAM_SET values are important because they dictate the search order that RUV uses to determine rule matching and selection processes.
Sample Construction of a Rule Set

This section provides a sample of how to construct a rule set that will cause batch journaling and logging of VSAM files. You will modify the rule set example (see Figure 30) to match with your batch journaling requirements. Perform the following process to create a sample rule set:

1. Print and examine the example JCL found in the $DEFAULT member of the RUV.CNTL library (see Figure 30). $DEFAULT will be in the repository if you completed the installation IVF tasks. The following command set will print $DEFAULT:

   ```
   SET REPORT(DETAIL);
   REPORT RULE_SET($DEFAULT)
   JOURNAL_MODEL($DEFAULT)
   INTERNAL_READER($DEFAULT)
   JOB_JCL($DEFAULT);
   ```

   For other rule set examples, print and examine the sample VSAM_SET and JOB_SET members that are provided in the $SAMPLE member of the RUV.CNTL library. Print them with the following command set:

   ```
   SET REPORT(DETAIL);
   REPORT VSAM_SET($SAMPLE);
   REPORT JOB_SET($SAMPLE);
   ```

---

**Figure 30  Sample Rule Set $DEFAULT (Part 1 of 2)**

```
2. Create a copy of member $SAMPLE of the RUV.CNTL library as shown in Figure 30 on page 194. Change all of the $SAMPLE labels to the 32-character labels of your choice. In the example, use element_TESTING, where element is for a JOB, VSAM, or RULE set label. Modify the optional comments as appropriate.

3. Under the ADD VSAM_SET section, find VSAM_RULE and replace NAME.VSAM.DSN with the name of the VSAM file to be journaled (TST.RUV.SAMPLE). The VSAM rule also uses the parameter BOTH to specify that both before and after-images are to be recorded for this file.

4. Under the ADD JOB_SET section, find JOB_RULE and replace YOURJOB with TST*. The asterisk character ‘*’ is a wildcard specification character that allows you to cover many jobs with a single rule.

   Also under the ADD JOB_SET section in, find JOB_RULE and replace the program name YOURPGM with TESTAPP. (TESTAPP is the name of the application that uses the TST.RUV.SAMPLE file.)

   This job rule accepts the default site specifications for journaling output and recovery JCL. It also requests that the job be allowed to continue if an error occurs while journaling by using the JOURNAL_ABEND(CONTINUE) keyword.

5. After the $DEFAULT entry under the ADD RULE_SET section, replace the comment lines as indicated with the JOB_SET and VSAM_SET 32-character labels you chose in procedure 2.

6. Before you use it, store the completed sample in the repository. The JCL shown in Figure 31 on page 196 will save the new rules in the repository.

7. Activate and test the rule set you have just built to verify its structure and flow. See “Activating a Rule Set” on page 198 and “Testing a Rule Set” on page 201.
The rule set construction sample process you used is a beginning to developing comprehensive and more complex rule sets. Remember to develop the rule sets in a logical and orderly manner.

The labeling scheme that you develop for your rule set names is important and should be worked out independently of the coding process. Creating the 32-character rule set labels connects various elements together. By learning the connection between elements, you will find reusing the names to be a logical and convenient process.
Figure 32 illustrates the recommended general order of the rule set coding process. It also illustrates the need to carefully label elements as they are created and reused in the rule set construction process.

**Figure 32  Rule Set Build and Activation Flow**

1. **ADD JOURNAL_MODEL** (JOURNAL_TESTING)
   ```sql
   QSAM"(USER_DSN_NAME,
   "SPACE=(CYL(500,250)"
   "UNIT=SYSDA,DISP=(,CATLG)"
   ```
   Do this to define the data set that will be used to journal batch jobs. Carefully select meaningful labels for your rule set keyword values.

2. **ADD JOB_SET** (JOB_TESTING)
   ```sql
   JOB_RULE(PAYREG*,*
   JOURNAL(JOURNAL_TESTING)
   ```
   Do this to run RUV batch journaling for all jobs that begin with PAYREG. Notice that the JOB_SET_JOURNAL_TESTING label came from the journal model label.

3. **ADD VSAM_SET** (VSAM_TESTING)
   ```sql
   VSAM_RULE(PAY*,JOURNAL)
   ```
   Do this to journal all VSAM files that begin with PAY.

4. **ADD RULE_SET** (RULE_TESTING)
   ```sql
   JOB_SET(JOB_TESTING)
   VSAM_SET(VSAM_TESTING)
   ```
   Do this to build a rule. When active, this rule set definition states that JOB_TESTING and VSAM_TESTING cause journaling as defined by their parameters.

5. **ACTIVATE RULE_SET** (RULE_TESTING)
   Store rule set RULE_TESTING in the FUV repository until it is activated

6. **Activate rule set RULE_TESTING when appropriate.**
Activating a Rule Set

You must activate a rule set before RUV can use it. RUV supports two levels of activation, subsystem-level and job-level. The subsystem-level rule set controls all jobs running in the z/OS system. A job-level rule set applies to a single batch job only. Only one rule set may be active at the subsystem level, but each batch job may have its own unique rule set.

Normal RUV operation is best controlled by a subsystem-level rule set. Job-level rule sets should be used for testing purposes and special situations that may require you to override the system-level rule set. It is suggested that you test all changes to rule sets before activating them at the subsystem level.

After you activate a rule set, RUV searches the rule set when a VSAM file is opened to determine the action to take (if any action is needed).

NOTE
You must reactivate your rule set after you update any user variables.

Using the ACTIVATE RULE_SET Command

Job-Level Rule Sets

To active a job-level rule set, execute the RUVZSM0 program with the ACTIVATE RULE_SET command, as shown in the following example. Use the value of the RULE_SET keyword to define the name of the rule set to activate. (This name is the same name you use as the value of the RULE_SET keyword on an ADD RULE_SET statement.)

ACTIVATE RULE_SET(TESTING);

System-Level Rule Sets

You can activate the subsystem-level rule set only with the ISPF interface. For more information, see “Using the ISPF Interface to Work with Rule Sets” on page 205.
Using the TRACE_ON Keyword

You can include the optional TRACE_ON keyword on the ACTIVATE RULE_SET command to view rules as you test them with the TRY command. The TRACE_ON keyword has no values. For more information about the TRY command, see “Testing a Rule Set” on page 201.

Processing the ACTIVATE RULE_SET Command

RUV establishes an association between a batch job and a rule set whenever one of the rules in the rule set is matched. Once RUV establishes this association, the job completes under the same rule sets that it started under.

In response to the ACTIVATE RULE_SET command, RUV reads the named rule set from the RUV repository and loads into memory the user variables and the rules that correspond to the named VSAM sets and job sets. RUV loads into memory the internal reader and job JCL entries at the beginning of a job step when the rule/job association is made. RUV reads journal models as they are needed to allocate journals or logs. See Table 10.

Table 10  Rule Set Processing

<table>
<thead>
<tr>
<th>Rule Set Entry</th>
<th>Loaded into Memory</th>
<th>RUV Variables Expanded</th>
<th>Action If Not in Repository</th>
<th>Changes to Entry in Repository Take Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSAM set</td>
<td>during ACTIVATE command</td>
<td>no</td>
<td>skipped</td>
<td>during next ACTIVATE command</td>
</tr>
<tr>
<td>job set</td>
<td>during ACTIVATE command</td>
<td>no</td>
<td>skipped</td>
<td>during next ACTIVATE command</td>
</tr>
<tr>
<td>user variable set</td>
<td>during ACTIVATE command</td>
<td>no</td>
<td>skipped</td>
<td>during next ACTIVATE command</td>
</tr>
<tr>
<td>internal reader</td>
<td>beginning of job step when interest in a VSAM file has been established</td>
<td>yes</td>
<td>search for $DEFAULT</td>
<td>beginning of next job step after interest in a VSAM file has been established</td>
</tr>
<tr>
<td>skeleton JCL</td>
<td>beginning of job step when interest in a VSAM file has been established</td>
<td>yes</td>
<td>search for $DEFAULT</td>
<td>beginning of next job step after interest in a VSAM file has been established</td>
</tr>
<tr>
<td>journal model</td>
<td>at journal or log allocation time</td>
<td>yes</td>
<td>search for $DEFAULT</td>
<td>at next use</td>
</tr>
<tr>
<td>external vendor</td>
<td>during ACTIVATE command</td>
<td>no</td>
<td>skipped</td>
<td>during next ACTIVATE command</td>
</tr>
</tbody>
</table>
Coding the ACTIVATE RULE_SET Statement

To code an ACTIVATE RULE_SET statement, use the following syntax.

```
NOTE
This command activates a job-level rule set. You activate a subsystem-level rule set from the ISPF interface or from a z/OS console.

ACTIVATE RULE_SET(name32)
   { TRACE_ON }
;
```
Testing a Rule Set

You can test a rule set in a batch environment by using the TRY command. You should always test a rule set before using it in a production system or activating it at the subsystem level. You can also obtain detailed information about the test results.

Coding TRY Statements

To code a TRY statement, use the following syntax:

```
TRY JOB(jobname) PROGRAM(program) DSN(dsname);
```

Using the TRY Command

Test a rule set by using the TRY command, as shown in the following example:

```
TRY JOB(PGNL010) PROGRAM(IDCAMS) DSN(GNLP.MASTER.ACCNTS);
TRY JOB(PGNL010) PROGRAM(INV987) DSN(GNLP.MASTER.ACCNTS);
TRY JOB(MGNL010) PROGRAM(INV987) DSN(GNLP.MASTER.ACCNTS);
TRY JOB(MGNL010) PROGRAM(INV100) DSN(GNLP.MASTER.ACCNTS);
TRY JOB(MGNL010) PROGRAM(INV200) DSN(GNLP.EXPENSE.REPORT);
```

Using the JOB, PROGRAM, and DSN Keywords

Use the JOB, PROGRAM, and DSN keywords to identify the particular job, program, and data set name to test. The values of these keywords are, respectively, the job name, the program name, and the data set name.

Obtaining Detailed Test Results

You can add the TRACE_ON keyword to the ACTIVATE RULE_SET command to obtain a detailed trace of all rules that RUV examines and the decisions about whether to bypass, accept, or reject the rules. You can control the amount of information produced by setting the appropriate reporting level (summary, detail, or full) with the SET REPORT command. For more information, see “Using the REPORT Keyword” on page 135.
Obtaining Detailed Test Results

Figure 33 shows an example of the results of the TRY command when the reporting
level is set to SUMMARY. Figure 34 shows an example of the results when the
reporting level is set to DETAIL.

Figure 35 is a test of the rule set and the results that display when the reporting level
is set to FULL. This rule set was built in the “Coding and Constructing a Rule Set”
section. It provides you with a preview of what would happen if you execute the
KUPDATE program.

---

**Figure 33** TRY Command Summary-Level Results

```
353 ACTIVATE RULE_SET(V11 TEST)
354 TRACE_ON
355 ;
RUV202177I RUV JOB LEVEL RULE SET: V11TEST
RUV202177I RUV SYSTEM LEVEL RULE SET: $EXCLUDE_ALL
356 SET REPORT(SUMMARY);
357 TRY JOB(ISIRAPZU) PROGRAM(ZZUKSDS)
```

---

**Figure 34** TRY Command Detail-Level Results (Part 1 of 2)

```
356 ACTIVATE RULE_SET(V11 TEST)
357 TRACE_ON
358 ;
RUV202177I RUV JOB LEVEL RULE SET: V11TEST
RUV202177I RUV SYSTEM LEVEL RULE SET: $EXCLUDE_ALL
359 SET REPORT(DETAIL);
360 TRY JOB(ISIRAPZU) PROGRAM(ZZUKSDS)
```

---
Figure 34  TRY Command Detail-Level Results (Part 2 of 2)

```plaintext
356 ACTIVATE RULE_SET(V11 TEST)
357 TRACE_ON
358:
RUV202177I RUV JOB LEVEL RULE SET: V11TEST
RUV202177I RUV SYSTEM LEVEL RULE SET: $EXCLUDE_ALL
359 SET REPORT(FULL):
360 TRY JOB(ISIRAPZU) PROGRAM(ZZUKSDS)
```

```plaintext
JOB LEVEL RULE SET
  VSAM: RUVRUV.QA*
  MATCHED VSAM: ISIRAM.*
  BACKOUT JOURNALING FOR DSNAME: ISIRAM.RUV.TEST.KSDS
  FORWARD JOURNALING FOR DSNAME: ISIRAM.RUV.TEST.KSDS
```

Figure 35  TRY Command Full-Level Results (Part 1 of 2)

```plaintext
356 ACTIVATE RULE_SET(V11 TEST)
357 TRACE_ON
358:
RUV202177I RUV JOB LEVEL RULE SET: V11TEST
RUV202177I RUV SYSTEM LEVEL RULE SET: $EXCLUDE_ALL
359 SET REPORT(FULL):
360 TRY JOB(ISIRAPZU) PROGRAM(ZZUKSDS)
```

```plaintext
JOB LEVEL RULE SET
  JOB: *        PROGRAM: DFH*
  JOB: *        PROGRAM: DFS*
  JOB: *        PROGRAM: DXP*
  JOB: *        PROGRAM: VHC*
  JOB: *        PROGRAM: DSN*
  JOB: *        PROGRAM: ARC*
  JOB: *        PROGRAM: GIM*
  JOB: *        PROGRAM: RUVZSM0
  JOB: *        PROGRAM: DFSSORT
  JOB: *        PROGRAM: ISIDRS
  JOB: *        PROGRAM: RPCMAIN
  JOB: *        PROGRAM: RCMBP0G
  JOB: *        PROGRAM: ISICFM
  JOB: *        PROGRAM: ARCCTL
  JOB: *        PROGRAM: IEFIC
  JOB: *        PROGRAM: S200MAI
  JOB: *        PROGRAM: RUV*
  MATCHED JOB: TST*        PROGRAM: KUPDATE
                      JOURNAL ($DEFAULT)
                      JOURNAL_ABEND (CONTINUE)
                      JOB_ABEND (SUBMIT)
                      JOB_JCL ($DEFAULT)
                      INTERNAL_READER ($DEFAULT)
  JOB LEVEL RULE SET
  MATCHED VSAM: *
  EXCLUDED DSNAME: TST.RUV.SAMPLE
RUV202420I RUV BCSS USING JOB LEVEL RULE SET: TESTING
RUV202420I RUV BCSS USING SYSTEM LEVEL RULE SET: RUV_TEST
RUV202421I RUV BCSS SELECTED BY JOB MASK: TST* PROGRAM MASK: KUPDATE
RUV202430I RUV BCSS LOG AND JOURNAL MODEL OR DSNAME ARE THE SAME; BLENDING IN
          EFFECT.
RUV202420I RUV BCSS LOG MODEL: $DEFAULT PROTECTION LEVEL 1
RUV202425I RUV BCSS // DD DSN=RUVRUV.D2009061.T2139403.DEFAULT,
RUV202425I RUV BCSS // SPACE=(CYL,(15,5),RLSE),
RUV202425I RUV BCSS // DISP=(MOD,DELETE,CATLG),
RUV202425I RUV BCSS // UNIT=WORK
RUV202426I RUV BCSS ALLOCATED LOG DDNAME: SIMLOG DSNAME:
RUVRUV.D2009061.T2139403.DEFAULT
RUV202420I RUV BCSS JOB_JCL MODEL: $DEFAULT
RUV202471I RUV BCSS BACKOUT JCL WILL BE SUBMITTED TO JES CLASS A.
RUV202420I RUV BCSS SUPPORT FOR X37 ON QSAM LOG WAS NOT REQUESTED
```
The TRY command generates journal, log, and internal reader allocation JCL and recovery JCL. It allows you to view the search of the rule set and the results of the rule set. Figure 35 on page 203 shows each job rule as it is being tested. When a matching rule is found, the complete rule is displayed. Journal and log allocations are simulated, and internal reader and recovery JCL expansions are also performed and displayed. This report provides you with a simulated picture of what will happen when the job actually runs.

It also shows errors that would occur. The most common error is undefined user variables. Look for the following error message:

Review the generated JCL to determine if it is what you want. You can see that additional tailoring is needed for JOB_JCL($DEFAULT). The STEPLIB is invalid, consequently, you need to correct the default recovery JCL or select another JOB_JCL template.

You may wish to customize other parts of the rule set (JOB_JCL, JOURNAL_MODEL, INTERNAL_READER) in order to achieve the precise protection you desire.
Using the ISPF Interface to Work with Rule Sets

You can use the ISPF interface to work with rule sets.

Accessing the Subsystem Detail Panel

To access rule sets through the ISPF interface, perform the following steps:

1. On the RUV Options panel, as described in “Starting the RUV ISPF Interface” on page 113, select the option labeled Subsystem / Rule Set Information. RUV displays the Subsystem Detail panel (Figure 36).

The Subsystem Detail panel provides access to rule sets. It also provides information about the RUV product, the operating system and the activity details for the subsystem connected to this RUV session. If you have update authority, you can:

- select Backup to enable or disable the Backup-While-Open feature
- select Logging to enable or disable the Batch Journaling Facility

Command ===> ____________________________________________________________________
F1=Help      F3=Exit     F10=Actions  F12=Cancel
On the Subsystem Detail panel (Figure 36) enter any character (letters is shown in example) in the Show field, and press Enter. RUV displays the Ruleset List panel that lists the rule sets that are associated with the current SSID (Figure 37).

You can use this panel to select a rule set or perform an action on the rule set.

- To create, edit, or delete a rule set, your logon ID must have update mode access.
- To activate a rule set, your logon ID must have command mode access. Access modes are governed by your security system, such as RACF.
Selecting a Type of Rule Set

To select one of the different types of rule set options, perform the following steps:

1. On the Ruleset List panel (Figure 37), enter any character (letter x is shown in example) in the Show field, and press Enter. RUV displays the Show popup window (Figure 38).

2. On the popup window, enter the number corresponding with the rule set option you want to list, and press Enter. RUV modifies the list on the Ruleset List panel to show the appropriate rule sets.

Figure 38  Show Popup Window

You may select any rule set in the list and take action on it.
Selecting a Rule Set for Action

To select a rule set and perform an action on it, perform the following steps:

1. Type any character in the selection field to the left of the desired rule set name. Do not press Enter.

2. Select the File action on the action bar, and press Enter. The pull-down File menu list of options is displayed (see Figure 39).

Figure 39  File Menu Options on Action Bar
On the selection field of the File action pull-down list, type the number that corresponds to the one of the following choices:

1. **(New)** Create a new rule. RUV displays the RUV Ruleset Templates panel, see “Creating a New Rule Set” on page 211.

2. **(Edit)** Update the selected rule. RUV displays the rule in an ISPF edit panel. You can modify the rule as needed.

   The rule set is displayed in a TSO window for review or editing, as shown in Figure 40.

   **Figure 40  Ruleset in ISPF Edit Panel**

   ![Ruleset in ISPF Edit Panel](image)

   Use standard ISPF Edit conventions to modify the data set as needed.

   Because the editing is controlled by ISPF and is outside of RUV, you must type END on the command line to save the changes and return to RUV.

   When you return to RUV, enter the END command (F3) to update the modified rule in the repository.

3. **(Delete)** Delete the selected rule. RUV displays messages indicating that the record was deleted.
4  (Activate) Activate the selected rule.

When you select option 4. Activate, the panel shown in Figure 41 is displayed.

**Figure 41  Activate the Selected Rule**

![Ruleset List Panel]

From this panel, you can activate the selected rule (by pressing F10) or cancel the action (by pressing F12).

**Figure 42** shows the PF key assignments for this panel.

**Figure 42  Key Assignments**

<table>
<thead>
<tr>
<th>Key</th>
<th>Definition</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>HELP</td>
<td>NO</td>
<td>Help</td>
</tr>
<tr>
<td>F2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F10</td>
<td>END</td>
<td>SHORT</td>
<td>OK</td>
</tr>
<tr>
<td>F11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F12</td>
<td>CANCEL</td>
<td>SHORT</td>
<td>Cancel</td>
</tr>
</tbody>
</table>

210 Recovery Utility for VSAM User Guide
Creating a New Rule Set

RUV provides a set of templates for rule sets. To access the templates provided by RUV, perform the following steps.

1. On the pull-down File menu list of options (see Figure 39), enter the number 1 to create a new rule set, end press Enter. The RUV Ruleset/Group/Model Templates panel (Figure 43) is displayed.

   Figure 43   RUV Ruleset/Group/Model Templates Panel

   Enter number for template type.

   RUVU010    RUV Ruleset/Group/Model Templates
   Option ===>

   Enter number for template type.

   ###  Template Type
   R1  Rule_Set
   R2  VSAM_Set
   R3  Job_Set
   R4  Job_JCL
   R5  Journal_Model
   R6  User_Variables
   R7  Internal_Reader
   R8  External_Vendor
   R9  VSAM_Group_Definition
   R10  Output_Model
   R11  VSAM_Name_Model

   Enter END command to cancel MODEL command.

2. In the Option field, enter the number that corresponds with the type of rule set that you want to create and press Enter. RUV invokes ISPF Edit with the MODEL command (Figure 40).

3. Use the standard ISPF editing conventions to modify the template as needed. Name or rename the file, and save all changes before exiting. Enter the END command on the command line to save the changes and return to RUV.

4. When you return to RUV, enter the END command (F3) to update the modified rule set in the repository.
Deleting Rule Set Elements

You can delete individual rule set elements (records) from the repository. To remove the records, use the DELETE command.

**WARNING**
The DELETE command permanently removes the record from the repository. You may want to be sure you have a recent backup of the repository before you execute DELETE commands.

Other sections in this manual describe using the DELETE command to remove other types of records (such as VSAM file and archive file records) from the repository. See the index for specific page numbers.

Using the DELETE Command

Use the DELETE command as shown in the following example:

```
DELETE JOB_JCL(TESTJOB*, TSTJ*);
```

The command verb is followed by a keyword that identifies the type of record to delete.

Using DELETE Command Keywords

You can use one of the following keywords on each DELETE command to delete the indicated type of record:

- INTERNAL_READER
- JOB_JCL
- JOB_SET
- JOURNAL_MODEL
- RULE_SET
- USER_VARIABLES
- VSAM_SET
- EXTERNAL_VENDOR
- OUTPUT_MODEL
- VSAM_GROUP_DEFINITION
- VSAM_NAME_MODEL
Using DELETE Command Keyword Values

The value of each keyword specifies which record (or records) to delete, as identified by the user-defined name of the record (name32). You can use masking (wildcard) characters in the name to select multiple records for deletion, and you can include multiple names in the list.

Coding DELETE Statements

To code a DELETE statement, use the following syntax:

```plaintext
DELETE INTERNAL_READER(name32, *, ... ) ;
DELETE JOB_JCL(name32, *, ... ) ;
DELETE JOB_SET(name32, *, ... ) ;
DELETE JOURNAL_MODEL(name32, *, ... ) ;
DELETE RULE_SET(name32, *, ... ) ;
DELETE USER_VARIABLES(name32, *, ... ) ;
DELETE VSAM_SET(name32, *, ... ) ;
DELETE EXTERNAL_VENDOR(name32, *, ... ) ;
DELETE OUTPUT_MODEL(name32, *, ... ) ;
DELETE VSAM_GROUP_DEFINITION(name32, *, ... ) ;
DELETE VSAM_GROUP(name32, *, ... ) ;
DELETE VSAM_NAME_MODEL(name32, *, ... ) ;
```
Chapter 5  Using RUV in a CICS-Only Environment 215

This chapter discusses the use of RUV in a CICS-only environment. The following information is included:

Introducing RUV in a CICS-only Environment ........................................ 217
  Archive Requirements .............................................................. 217
  Backup Requirements ............................................................. 221
  Controlling the Backup Subsystem (RUVM). .............................. 221
  Daylight Savings Time (DST) Considerations ............................. 222
Performing a Recovery in a CICS-Only Environment ...................... 224
  Performing a Forward Recovery ............................................... 224
  Perform a Backout Recovery .................................................... 226
Archiving CICS Journals .............................................................. 227
  Using the ARCHIVE JOURNAL_IN Command ........................... 227
  Using the JOURNAL_IN Keyword .......................................... 228
  Using the APPLID Keyword ................................................... 228
  Using the COMPLETE_COPY Keyword .................................. 228
  Using the ARCHIVE_OUT Keyword ....................................... 228
  Using Optional Output Keywords ............................................ 229
  Coding ARCHIVE JOURNAL_IN Statements ............................ 229
Archiving z/OS Log Streams .......................................................... 230
  Using the ARCHIVE LOG_STREAM_IN Command .................... 230
  Using the LOG_STREAM_IN Keyword ...................................... 230
  Using the VIEW Keyword ..................................................... 231
  Using Optional Output Keywords ............................................ 233
  Using the DELETE Keyword .................................................. 231
  Using the COMPLETE_COPY Keyword .................................. 232
  Coding ARCHIVE LOG_STREAM_IN Statements ....................... 233
Printing Log Stream Data ......................................................... 234
  Using the PRINT LOG_STREAM_IN Command ......................... 234
  Using the LOG_STREAM_IN Keyword ..................................... 234
  Using the DELETE Keyword ............................................... 234
  Using the START_TIME Keyword ........................................... 234
  Coding PRINT LOG_STREAM_IN Statements ............................. 235
Managing Log Stream Records .................................................. 236
  Using the ADD LOG_STREAM Command .................................... 236
  Using the UPDATE LOG_STREAM Command ................................ 236
  Using the DELETE LOG_STREAM Command ............................... 237
  Using the REPORT LOG_STREAM Command ............................... 237
  Using the LOG_STREAM Keyword ......................................... 237
  Using Optional Keywords .................................................. 238
  Coding ADD LOG_STREAM Statements ...................................... 238
  Coding UPDATE LOG_STREAM Statements ............................... 238
  Coding DELETE LOG_STREAM Statements ............................... 238
Managing the Transaction Server Log of Logs ............................. 239
  Using the REGISTER LOG_OF_LOGS Command ......................... 239
  Using the ADD LOG_OF_LOGS Command ................................... 240
  Using the UPDATE LOG_OF_LOGS Command ............................. 240
  Using the DELETE LOG_OF_LOGS Command ............................. 240
  Using the REPORT LOG_OF_LOGS Command ............................. 241
  Using the LOG_OF_LOGS Keyword ......................................... 241
  Using the DELETE Keyword ................................................ 241
  Using the DAYS Keyword .................................................. 242
  Using the START_TIME Keyword ........................................... 242
  Using Optional Keywords .................................................. 242
  Coding ADD LOG_OF_LOGS Statements ................................. 242
  Coding DELETE LOG_OF_LOGS Statements ............................. 243
  Coding REGISTER LOG_OF_LOGS Statements .......................... 243
  Coding UPDATE LOG_OF_LOGS Statements ............................. 243
Introducing RUV in a CICS-only Environment

RUV can recover VSAM files that are updated by CICS or Transaction Server. RUV supports and recovers the following file structures and organizations:

- VSAM key-sequenced data sets (KSDS)
- VSAM entry-sequenced data sets (ESDS)
- VSAM record-level-sharing data sets (RLS)
- VSAM fixed length relative record data sets (RRDS)
- VSAM variable length RRDS (for Transaction Server 1.3 and later)

CICS does not have to be installed or running when you perform a recovery. VSAM record-level-sharing files are recovered in non-RLS mode; however, RLS locks may be retained if you want. For more information, see Chapter 11, “Recovering VSAM Data Sets.”

RUV identifies VSAM data sets by data set name. RUV automatically recognizes path and base associations. RUV recovers the complete VSAM sphere.

Archive Requirements

RUV recovers CICS VSAM files by using the recovery information written by CICS to CICS journals and log streams. CICS-updatable VSAM files are identified in the CICS file definitions. The CICS file definitions specify what type of recovery information is to be recorded for the VSAM data set (forward or backout recovery) and where the information is to be written. At file open time, CICS writes tie-up records that associate the CICS file definition name with the VSAM DSN. Tie-up records are written to the recovery journal and, for Transaction Server, the log of logs. For the Transaction Server environment, registration of the log of logs is recommended before archiving the log stream. For more information, see “Using the REGISTER LOG_OF_LOGS Command” on page 239.

RUV uses the tie-up records to build and maintain a CICS file definition/DSN association. RUV records this association in the RUV repository. You can view this association by APPLID, FILEID, or VSAM DSN through the RUV ISPF interface.
Coding CICS File Definitions for the CICS Transaction Server Environment

You must modify the CICS file definitions and journal definitions to have CICS record the recovery information that you need. The parameters that you should define for the CICS TS file definitions are listed in Figure 44 on page 218 and described in Table 11 on page 219. These parameters apply to logging and journaling only and do not include all the parameters that you can define for the CICS file definitions. The remainder of the CICS file definitions are table definitions for managing VSAM journal records generated by CICS. For the full procedure, see the Recovery Utility for VSAM Installation and Customization Guide.

Figure 44  CICS File Definitions for CICS TS Environments

CEDA DEFINE FILE(fileid) GROUP(groupname)
.  
.  
FWDRECOVLOG(nn) <= nn - forward recovery journal
RECOVERY(ALL)
.  
.  

NOTE
To recover data sets by using the recovery utility, you must specify FWDRECOVLOG(nn) and RECOVERY(ALL).

In the CICS Transaction Server environment, you can define files by using Resource Definition Online (RDO). See the IBM CICS Transaction Server Resource Definition (Online) manual for information on using RDO.

RUV can archive forward and backout recovery when you specify the following parameters in the CICS file definitions:

- If the CICS file definitions specify the following parameters, the RUV ARCHIVE must include the system log and the user journal:
  - FWDRECOVLOG(nn)
  - RECOVERY(ALL)
If the CICS file definitions specify the following parameters, the RUV ARCHIVE must be run against the user journal:

- FWDRECOVLOG(NO)
- RECOVERY(NONE)
- JNLREAD(UPDATEONLY)
- JNLUPDATE(YES)
- JNLADD(ALL)
- JOURNAL(nn)

If the CICS file definitions specify the following parameters, the RUV ARCHIVE must include the system log (DFHJ01/A/B/X) and the user journal only:

- FWDRECOVLOG(nn)
- RECOVERY(ALL)
- JNLREAD(UPDATEONLY)
- JNLUPDATE(YES)
- JNLADD(ALL)
- JOURNAL(nn)

Table 11  CICS File Definition Parameters for CICS TS Environments (Part 1 of 2)

<table>
<thead>
<tr>
<th>CICS File Definition Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE(fileid)</td>
<td>Identifies the file defined in this CICS file definition. The name must be from 1 to 8 characters long. The recovery utility uses this file name to correlate a data set and the corresponding journal records during a recovery.</td>
</tr>
<tr>
<td>FWDRECOVLOG(NO</td>
<td>nn)</td>
</tr>
<tr>
<td>JNLADD(ALL)</td>
<td>Defines the ADD operations you want recorded on the journal specified by the JOURNAL parameter. ALL journals the file control WRITE operation before and after the VSAM I/O operation.</td>
</tr>
<tr>
<td>JNLREAD(UPDATEONLY)</td>
<td>Defines the read operations you want recorded on the journal specified by the JOURNAL parameter. UPDATEONLY journals READ UPDATE operations only.</td>
</tr>
<tr>
<td>JNLUPDATE(YES)</td>
<td>Specifies that you want REWRITE and DELETE operations recorded on the journal specified by the JOURNAL parameter.</td>
</tr>
</tbody>
</table>
CICS 3.3 and 4.x Recovery Environments

CICS records recovery information to user journals. Typically, these user journals are organized as two sequential extents on disk. The sequential extents are referred to as the A/B extents. When an extent fills, it is closed and CICS opens the other extent for output. You need to save the recovery information that was recorded in the closed extent. You do this by using the ARCHIVE JOURNAL_IN command. A/B extent switching is time-sensitive; the archive must be completed before the extent is reused for output or you will lose valuable recovery information. If you experience numerous A/B extent switches during the day, consider using the ARCHIVE ACCUM command to consolidate multiple archives into a daily archive.

If you archive DFHJ01 A and B extents, you should also archive DFHJ01X after an emergency restart.

(See note on page 36 concerning journal support for unsupported CICS releases.)

Transaction Server Recovery Environments

Transactions Server records recovery information to z/OS log streams. Log streams do not switch extents, so archiving of log streams is not time-sensitive. Log stream archives may be run at your convenience or as needed.

NOTE

It is your responsibility to archive required recovery information from CICS A/B/X extents or log streams before you run a recovery.
Batch Recovery Environment

RUVCAN journal updates made to VSAM files by programs running in batch. If you’re only interested in CICS recovery, you should disable the Batch Journaling Facility or use an EXCLUDE_ALL rule set. For an example of an EXCLUDE_ALL rule set, see member #DEFAULT in the RUVCNTL library.

Backup Requirements

You need a backup plan in place to periodically copy files to a backup. RUVCAN take backups of VSAM files. The RUVCAN Backup-While-Open feature (BWO) is of particular interest in CICS-only environments. BWO allows you to back up a VSAM file while it is being updated in CICS. RUVCAN recovery can also use SMS backups.

Controlling the Backup Subsystem (RUVM)

RUVCAN provides a command interface you can use to control the BWO Subsystems from a CICS region. The command interface can be executed as a CICS transaction from a CICS terminal or as a CICS console using the z/OS MODIFY command.

CICS recognizes the RUVM transaction and commands (Table 12) followed by at least one blank space (for example, STATUS).

Table 12 RUVM CICS z/OS Console Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Issued from...</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>F cicsjob,RUVM FORCE</code></td>
<td><code>z/OS console</code></td>
<td>Terminates all subtasks controlled by the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td><code>F cicsjob,RUVM MONITOR</code></td>
<td><code>CICS terminal</code></td>
<td>Starts a CICS task that displays at the CICS terminal the status of all subtasks controlled by the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td><code>F cicsjob,RUVM RESTART</code></td>
<td><code>z/OS console</code></td>
<td>Performs a shutdown and then a start of the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td><code>F cicsjob,RUVM START</code></td>
<td><code>CICS terminal</code></td>
<td>Starts the Backup control task within CICS that communicates with the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td><code>F cicsjob,RUVM STATUS</code></td>
<td><code>z/OS console</code></td>
<td>Displays the status of any active subtasks performing backups for a Batch Backup job.</td>
</tr>
<tr>
<td><code>F cicsjob,RUVM SHUTDOWN</code></td>
<td><code>CICS terminal</code></td>
<td>Shuts down the RUVM BWO Subsystem.</td>
</tr>
</tbody>
</table>
Daylight Savings Time (DST) Considerations

If you reset your system clock back one hour to Fall daylight savings time (DST), an overlap in time ranges can occur during that period of time. This overlap may cause erroneous processing of data. Special considerations are necessary to manage CICS data and batch journal data that are produced following a time reset.

CICS records VSAM file updates with a local timestamp. RUV uses GMT timestamps (via an STCK command) for batch journals and registration of RUV backups and archives. To address this difference, RUV converts the CICS local timestamp to a GMT timestamp when a CICS journal is archived. It does this by using fields in the CVT to calculate the difference between local time and GMT. CICS then stores recovery information with a GMT timestamp.

RUV does not need to perform the time conversions for Transaction Server because Transaction Server provides time in both local and GMT formats.

**NOTE**

It is recommended that you use the system operator ‘SET DATE=YYY.DDD,CLOCK=HH.MM.SS’ commands to adjust the CVT time difference values rather than changing the system clock.

**Reset Verification Procedure for Daylight Savings Time**

For the CICS data that is created during DST, the user should ensure that RUV archives the CICS journal data at the point before you reset the time to go back one hour. By performing the following procedure, you can ensure that RUV will recover the data:

1. Stop CICS.
2. Archive journals.
3. Stop batch journaling.
4. Reset the clock.
5. Proceed with normal processing of CICS and batch jobs.
Recovery Verification Procedure for Daylight Savings Time

In the event a recover is required that covers this time fallback, the recover forward requires the following steps:

1. Restore the VSAM file from a backup.

2. Perform a recover forward using only the archives that were created up to the point before the time reset.

3. Perform a recover forward using only the archives that were created after the time reset.

You may consider other methods appropriate to your environment that can be used to achieve an accurate recovery for the DST time reset.
Performing a Recovery in a CICS-Only Environment

The following sections describe procedures for performing forward recovery, completed unit-of-work recovery, and backout recovery in a CICS-only environment.

Performing a Forward Recovery

Perform the following steps to recover one or more VSAM spheres in a CICS-only environment:

1. Determine which CICS files need to be recovered.
2. If CICS is operating, close the files and flush the log stream buffer to capture all data.
3. Archive CICS extents or log streams as required. See “Archiving CICS Journals” on page 227 or “Archiving z/OS Log Streams” on page 230.
4. Prepare a JCL to run the forward recovery.
   - You can use the RUV ISPF interface to interactively prepare a JCL using the most current backup and allow RUV to select the required archive(s). See “Using the ISPF Interface for Recovery” on page 495
   - You can manually code the batch control statements for the JCL. You can specify the following information:
     - the sphere cluster name
     - the DSN of the most current backup
     - the DSNs of all archives written after the backup was created
5. Restore the files from backups.
   - If you are using non-RUV registered backup, you must restore the VSAM files from backups using the proper non-RUV tool.
   - If you are using RUV-registered backups the restores may be done in the same step as the recovery step. See “Restoring Files from Backups” on page 457
6. Run the recovery job, and check the results.
7. Make the files available to the CICS regions.
Performing a UOW Recovery

Perform the following steps to recover forward one or more VSAM spheres in a CICS-only environment using the unit-of-work (UOW) archives:

1. Determine which CICS files need to be recovered.

2. If CICS is operating, close the files and flush the log stream buffer to capture all data.

3. Archive the DFHLOG transaction server log and the user log streams as required. See “Minimum UOW Requirements” on page 468 or “Archiving z/OS Log Streams” on page 230.

4. Prepare a JCL to run the forward recovery.

   You can manually code the batch control statements for the JCL. You can specify the following information:
   
   — the STOP_TIME (required)
   — the sphere cluster name
   — the DSN of the most current backup
   — the DSNs of all archives, including the UOW archives, written after the backup was created

5. If you are using non-RUV registered backups, you must restore the VSAM files from backups using the proper non-RUV tool.

   If you are using RUV-registered backups, the restores may be done in the same step as the recovery step. See “Restoring Files from Backups” on page 457

6. Run the recovery job, and check the results. During the UOW recovery, RUV manages the in-flight transaction. See “Unit-of-Work Recovery Operation” on page 43.

7. Make the files available to the CICS regions.

8. After the UOW recovery, restart CICS with START=COLD.
Perform a Backout Recovery

Perform the following steps to back out changes to one or more VSAM spheres in a CICS-only environment:

1. Determine which CICS files need to be recovered.

2. If CICS is operating, close the files and flush the log stream buffer to capture all data.

3. Archive CICS extents or log streams as required. See “Archiving CICS Journals” on page 227 or “Archiving z/OS Log Streams” on page 230.

4. Prepare a JCL to run the backout recovery.
   - You can use the RUV ISPF interface to interactively prepare a JCL for a backout recovery and allow RUV to select the required archive(s). See “Using the ISPF Interface for Recovery” on page 495.
   - You can manually code the batch control statements for the JCL. You can specify the following information:
     - the START_TIME and STOP_TIME values for recovery to a point in time
     - the DSN of the VSAM group or individual VSAM file

5. Run the recovery job, and check the results.

6. Make the files available to the CICS regions.
Archiving CICS Journals

RUV can read recovery information from CICS CICS4.x journal extents and record the recovery information to one or more RUV archives. You must archive CICS extents each time they switch; otherwise, the information in them may be overlaid, creating an integrity exposure. You can add a step to the journal switch job to perform the RUV archive.

For information about archiving other journal formats, see “Archiving z/OS Log Streams” on page 230, “Archiving RPCV Journals” on page 272, and “Recovering an RUV Repository Using SMF Data” on page 568.

Also see the note on page 36 concerning journal support for unsupported CICS releases.

**NOTE**
The JOURNAL MANAGER PLUS product from BMC Software can submit a user-defined archive job every time the CICS journal switches and can prevent the extents from being overwritten until the archive job completes successfully. For more information, see the JOURNAL MANAGER PLUS documentation set.

You must supply an APPLID value since CICS extent does not contain this data. RUV allows you to make more than one copy of the data with a single pass of the input.

For RUV to process and build the archive file correctly, input that is coming from the IBM DFHJUP utility into RUV must be defined as RECFM=VB and have the LOCAL option specified on the SUBSYS parameter. For information about batch processing of log stream data and about valid subsystem parameters, refer to the IBM documentation about CICS Transaction Server operations and utilities.

**NOTE**
The default time on the output file from the DFHJUP utility is provided in GMT. To specify the local time as the time on the output file, use the LOCAL keyword option.

Using the ARCHIVE JOURNAL_IN Command

To archive a CICS extent, use the ARCHIVE JOURNAL_IN command, as shown in the following example.

```
ARCHIVE JOURNAL_IN(DFHJ05A) APPLID(C41P2)
  ARCHIVE_OUT(JOUT1)
  ARCHIVE_OUT(JOUT2);
```
Using the JOURNAL_IN Keyword

Use the required JOURNAL_IN keyword on the ARCHIVE JOURNAL_IN command to identify the source of the CICS data to archive. Set the value to the fully qualified data set name or to the ddname of the journal extent in the JCL.

Using the APPLID Keyword

Use the required APPLID keyword on the ARCHIVE JOURNAL_IN command to map the CICS APPLID to the corresponding data set name. Set the value to the APPLID.

Using the COMPLETE_COPY Keyword

You should use the COMPLETE_COPY keyword only at the request of Product Support. Use this optional keyword on the ARCHIVE JOURNAL_IN command to copy the data from the journal in its original format, as well as copying the recovery data in RUV format. Although RUV needs only the recovery data from the journal, copying all data can be of benefit in the problem analysis effort. Copying all data increases the output size two or more times over the output size of copying recovery data in RUV format only.

This keyword has no values. If it is present, RUV copies all original data. If it is absent, RUV copies only the recovery data in RUV format.

Using the ARCHIVE_OUT Keyword

Use the required ARCHIVE_OUT keyword on the ARCHIVE JOURNAL_IN command to identify the data set to contain the output archive data. Set the value to the ddname of the data set in the JCL. You can have RUV create multiple copies of the output archive by coding multiple ARCHIVE_OUT keywords. RUV registers all output archives in the repository.

NOTE
The APPLID must be supplied in the command. CICS does not supply this information.
Using the UNIT_OF_WORK_OUT Keyword

Use the optional UNIT_OF_WORK_OUT keyword on the ARCHIVE JOURNAL_IN command to identify the data set to contain the output UOW archive data. This keyword is required if the archive will be used for a UOW recovery. Set the value to the ddname of the data set in the JCL.

Using Optional Output Keywords

You can use the following optional keywords with the ARCHIVE_OUT and UNIT_OF_WORK_OUT keywords:

- COMMENT (see page 107)
- LOCATION (see page 110)
- STATUS (see page 112)
- MUST_COMPLETE (see page 110)

Code these keywords and their values after the ddname value but before the closing parenthesis for the ARCHIVE_OUT value. Only one archive copy should have an active status.

Coding ARCHIVE JOURNAL_IN Statements

To code an ARCHIVE JOURNAL_IN statement, use the following syntax:

```
ARCHIVE JOURNAL_IN(ddn8_or_dsn44)
   APPLID(applid8)
   { COMPLETE_COPY }
   ARCHIVE_OUT(ddn8
      { LOCATION(location_name8) }
      { COMMENT(comment_fields) }
      { STATUS(ACTIVE | INACTIVE) }
      { MUST_COMPLETE (YES | NO) }
   }
   { UNIT_OF_WORK_OUT(ddn8) }
      { COMMENT(comment_fields) }
      { LOCATION(location_name8) }
      { STATUS(ACTIVE | INACTIVE) }
      { MUST_COMPLETE (YES | NO) }
      { ARCHIVE_OUT(ddn8, ... ) }
   ;
```
Archiving z/OS Log Streams

RUV reads recovery information from Transaction Server log streams and records the recovery information to one or more RUV archives. All output archives are registered in the RUV repository.

You must archive z/OS log streams before they can be used for RUV recovery. If you use the RUV UOW feature, both the system log and the user journal (log stream) must be captured in separate RUV Archive requests. For UOW processing, you should use the AUTODELETE(NO) and RETPD(ddd) z/OS parameter to preserve system log data.

**NOTE**
To allow RUV to capture the necessary data from the system log, the RETPD(ddd) value must be 1 or greater.

RUV allows you to make more than one copy of the data with a single pass. RUV also assists you with log stream management by optionally deleting data that has been captured for archive. The log streams should be archived on a regular basis to reduce the amount of time required for a recovery.

Using the ARCHIVE LOG_STREAM_IN Command

To archive log streams, use the ARCHIVE LOG_STREAM_IN command, as shown in the following example.

```
ARCHIVE LOG_STREAM_IN(TS.J03)
ARCHIVE_OUT(ARCHOUT)
UNIT_OF_WORK_OUT(OUWOUT)
```

The user journal data on the named log stream will be directed to the ARCHIVE_OUT data set. The system log data for a UOW will be directed to UNIT_OF_WORK_OUT data set.

Using the LOG_STREAM_IN Keyword

Use the required LOG_STREAM_IN keyword on the ARCHIVE command to identify the z/OS log stream that contains the data to archive. Set the LOG_STREAM_IN value to the fully qualified log stream name.
Using the VIEW Keyword

Use the optional VIEW keyword on the ARCHIVE LOG_STREAM_IN command to select log stream data that has been deleted. Log stream records are retained based on the value of the z/OS RETPD(ddd) parameter. The dddd value is the number of days for which data is to be retained. A record is active unless it is deleted by an application.

Set one of the following values. The default is ACTIVE

**ACTIVE**
- archive only records that are not deleted

**ALL**
- archive both active and inactive records

The RUV VIEW keyword is compliant with the IBM log stream function. For more information, see the IBM log stream documentation.

Using the DELETE Keyword

Use the optional DELETE keyword on the ARCHIVE LOG_STREAM_IN command to control whether RUV deletes the input log stream data when the archive process completes successfully. Set one of the following values. The default value is NO.

**NO**
- Do not delete the input log stream data.

**YES**
- Delete the input log stream data when the archive process completes successfully. RUV deletes all blocks that were created by Transaction Server except for the last block. RUV leaves the last block in the log stream and records the block ID in the repository. It uses this information to check for GAP when the next archive runs.

When you are archiving the DFHLOG, specify the DELETE(NO) keyword. It is impossible to do an emergency restart when the keyword is set to DELETE(YES).
Using the GAP Keyword

Use the optional GAP keyword on the ARCHIVE LOG_STREAM_IN command to specify the action to be taken when RUV detects missing data.

Set one of the following values. The default value is STOP.

**STOP**
Stop processing the log stream. After you analyze the cause of the gap and determine that the process needs to continue, you can rerun the process with the WARN or IGNORE value.

**WARN**
Produce a message, generate return code 4, and continue processing.

**IGNORE**
Produce a message, generate return code 0, and continue processing.

Using the COMPLETE_COPY Keyword

Use the optional COMPLETE_COPY keyword on the ARCHIVE JOURNAL_IN command to copy the data from the journal in its original format, as well as copying the recovery data in RUV format. You should use the COMPLETE_COPY keyword only at the request of Product Support. See “Using the COMPLETE_COPY Keyword” on page 228.

Using the ARCHIVE_OUT Keyword

Use the required ARCHIVE_OUT keyword on the ARCHIVE LOG_STREAM_IN command to identify the data set to contain the output archive data. Set the value to the ddname of the data set in the JCL. You can have RUV create multiple copies of the output archive by coding multiple ARCHIVE_OUT keywords. RUV registers all output archives in the repository.

Using the UNIT_OF_WORK_OUT Keyword

Use the optional UNIT_OF_WORK_OUT keyword on the ARCHIVE LOG_STREAM_IN command to identify the data set to contain the output UOW archive data. This keyword is required if the archive will be used for a UOW recovery. Set the value to the ddname of the data set in the JCL.
Using Optional Output Keywords

You can use the following optional keywords with the ARCHIVE_OUT and UNIT_OF_WORK_OUT keywords:

- COMMENT (see page 107)
- LOCATION (see page 110)
- STATUS (see page 112)
- MUST_COMPLETE (see page 110)

Code these keywords and their values after the ddname value but before the closing parenthesis for the ARCHIVE_OUT value. Only one archive copy should have an active status.

Coding ARCHIVE LOG_STREAM_IN Statements

To code an ARCHIVE LOG_STREAM_IN statement, use the following syntax:

```
ARCHIVE LOG_STREAM_IN(lsn26)
  { VIEW(ALL | ACTIVE) }
  { DELETE (NO | YES) }  
  { GAP(STOP | WARN | IGNORE) } 
  { COMPLETE_COPY }   
  { VIEW(ACTIVE | ALL) } 
  ARCHIVE_OUT(ddn8)
    { COMMENT(comment_fields) }     
    { LOCATION(location_name8) }     
    { STATUS(ACTIVE | INACTIVE) }     
    { MUST_COMPLETE (YES | NO) }     
  )
  { UNIT_OF_WORK_OUT(ddn8) }    
    { COMMENT(comment_fields) }    
    { LOCATION(location_name8) }    
    { STATUS(ACTIVE | INACTIVE) }    
    { MUST_COMPLETE (YES | NO) }    
  )
```
You can use the RUVZSM0 utility to print the data in a log stream in hexadecimal format. Product Support might request this information.

Using the PRINT LOG_STREAM_IN Command

Use the PRINT LOG_STREAM_IN command, as shown in the following example, to print selected records from the log stream.

```
PRINT LOG_STREAM_IN(TS.J03)
   DELETE(YES)
   START_TIME(20092140535001)
;
```

Using the LOG_STREAM_IN Keyword

Use the required LOG_STREAM_IN keyword on the PRINT LOG_STREAM_IN command to identify the z/OS log stream that contains the data to print. Set the value to the fully qualified log stream name.

Using the DELETE Keyword

Use the optional DELETE keyword on the PRINT LOG_STREAM_IN command to control whether RUV deletes the input log stream data when the print process completes successfully. Set one of the following values. The default value is NO.

- **NO**  Do not delete the input log stream data.
- **YES** Delete the input log stream data when the print process completes successfully.

Using the START_TIME Keyword

The START_TIME keyword is available on some commands to specify the start (earliest) timestamp of the range of records to select for processing. See “Using the START_TIME Keyword” on page 111 for an explanation of this keyword.
## Coding PRINT LOG_STREAM_IN Statements

To code a PRINT LOG_STREAM_IN statement, use the following syntax:

```plaintext
PRINT LOG_STREAM_IN(lsname)
  { DELETE(NO | YES) }
  { START_TIME(yyyyMMddhhmmss) }
;
```
Managing Log Stream Records

RUU uses log stream records to track information. Each time a log stream is archived, RUV will record the date and time of the scan, job information, and the ending log stream block information (time and RBA). The ending log stream block information is used to reposition the log stream to the beginning point for the next archive operation. This minimizes archive time and eliminates duplication of data.

Using the ADD LOG_STREAM Command

To add a log stream record to the repository, you can use the ADD LOG_STREAM command, as shown in the following example.

```
ADD LOG_STREAM(TS.J03)
  COMMENT('TRANSACTION SERVER LOG STREAM 3')
  STATUS(INACTIVE)

NOTE
You can use START_TIME(2009) instead of the complete timestamp.
```

Using the UPDATE LOG_STREAM Command

To update the information in a log stream record, you can use the UPDATE LOG_STREAM command, as shown in the following example.

```
UPDATE LOG_STREAM(TS.J03)
  COMMENT('TRANSACTION SERVER LOG STREAM 3')
  START_TIME(20092140535001)
  STATUS(ACTIVE)
```
Using the **DELETE LOG_STREAM** Command

To delete a log stream record from the repository, you can use the DELETE LOG_STREAM command, as shown in the following example.

```plaintext
DELETE LOG_STREAM(TS.J*)
```

**Using the **REPORT LOG_STREAM** Command

To report the log stream record from the repository, use the REPORT LOG_STREAM command, as shown in the following example.

```plaintext
REPORT LOG_STREAM(TS.J*)
```

The REPORT LOG_STREAM output is described in “LOG_STREAM Report” on page 418.

**Using the **LOG_STREAM** Keyword

Use the LOG_STREAM keyword as follows to identify the log stream

- On the ADD LOG_STREAM or UPDATE LOG_STREAM command, set the value to the fully qualified log stream name. You can use masking (wildcard) characters with this command.

- On the DELETE LOG_STREAM command, set the value to the log stream name. You can use masking (wildcard) characters in the name, and you can include multiple names. You can provide multiple values. Separate the values with a command and a blank space.
Using Optional Keywords

You can use the following optional keywords on the ADD LOG_STREAM and UPDATE LOG_STREAM commands:

- COMMENT (see “Using the COMMENT Keyword” on page 107)
- STATUS (see “Using the STATUS Keyword” on page 112)
- START_TIME (see “Using the START_TIME Keyword” on page 111)

Coding ADD LOG_STREAM Statements

To code an ADD LOG_STREAM statement, use the following syntax:

```
ADD_LOG_STREAM(lsn26)
  { COMMENT(comment_fields) }
  { STATUS(ACTIVE | INACTIVE) }
;
```

Coding UPDATE LOG_STREAM Statements

To code an UPDATE LOG_STREAM statement, use the following syntax:

```
UPDATE_LOG_STREAM(lsn26)
  { COMMENT(comment_fields) }
  { STATUS(ACTIVE | INACTIVE) }
  { START_TIME(yyyyjjjhhmmssst) }
;
```

The START_TIME parameter resets the starting time for the next archive command. This is not necessary during normal operation.

Coding DELETE LOG_STREAM Statements

To code a DELETE LOG_STREAM statement, use the following syntax:

```
DELETE_LOG_STREAM(lsn26, *, ... )
;
```
Managing the Transaction Server Log of Logs

After the log_of_logs is registered, RUV can automatically track the relationship between the CICS APPLID/FILEID, the VSAM file DSN, and the archives by scanning the operation and storing the information in the LOG_OF_LOGS. The relationships are established by using the tie-up records produced by CICS and Transaction Server. You can use the Transaction Server log of logs to quickly build an inventory of VSAM files that are being updated by Transaction Server regions. RUV reads the log of logs log stream and records the DSN and APPLID_FILEID information in the RUV repository. This information is available with the REPORT LOG_OF_LOGS command as described in “LOG_OF_LOGS Report” on page 416.

You can view information about the APPLID/FILEID associations that RUV has built from the log of logs and has recorded in the repository by using the REPORT APPLID(*) and REPORT VSAM_FILE(*) commands. RUV does not record recovery data from the log of logs; it just uses the information to create APPLID/FILEID/DSN associations. These associations are also built from tie-up records that are recorded on log streams.

RUV does not use the log of logs during recovery processing but obtains the relevant information from the RUV repository. The log of logs is merely a fast, convenient method for populating the repository with this information.

You can use the DELETE LOG_OF_LOGS command to remove the repository reference to the log of logs and prevent automatic scanning of the log of logs. This reference cannot be removed automatically.

To suspend processing of the log of logs temporarily, you can change the status of the log of logs to INACTIVE.

The Last Scanned field on the Log of Logs List panel contains the date and time of the last archive of a log stream. This field is updated only when a log stream is archived. The Date and Time field on the Log of Logs Detail panel shows the date and time that the file was opened or closed by CICS and Transaction Server; this action may occur after the scan is performed.

Using the REGISTER LOG_OF_LOGS Command

To scan, register, and add all recoverable CICS VSAM files immediately, use the REGISTER LOG_OF_LOGS command, as shown in the following recommended method:

```sql
REGISTER LOG_OF_LOGS(PRODUCTION.LOL)
   DELETE(NO);
```
Using the ADD LOG_OF_LOGS Command

To automatically scan the Transaction Server’s log of logs each time a z/OS log stream is archived, add a log of logs record to the repository by using the ADD LOG_OF_LOGS command, as shown in the following example:

```
ADD LOG_OF_LOGS(PRODUCTION.LOL)
   COMMENT('Production log of logs')
   STATUSACTIVE);
```

Log of logs that are in an inactive status will not be automatically scanned. An automatic scan uses a default of DELETE(NO). The log of logs scan reads only new data that has been placed in the log since the last scan.

Using the UPDATE LOG_OF_LOGS Command

To update the COMMENT and STATUS information in the log of logs record in the repository, use the UPDATE LOG_OF_LOGS command, as shown in the following example:

```
UPDATE LOG_OF_LOGS(PRODUCTION.LOL)
   COMMENT('Production log of logs')
   STATUS(INACTIVE);
```

**NOTE**

No information other than the COMMENT and STATUS information can be updated with the UPDATE LOG_OF_LOGS command.

Using the DELETE LOG_OF_LOGS Command

To delete the log of logs record from the repository, use the DELETE LOG_OF_LOGS command, as shown in the following example:

```
DELETE LOG_OF_LOGS(P*.LOL)
;```
Using the **REPORT LOG_OF_LOGS** Command

To report the log of logs record from the repository, use the **REPORT LOG_OF_LOGS** command, as shown in the following example.

```
REPORT LOG_OF_LOGS(P*.LOL)
```

The **REPORT LOG_OF_LOGS** output is described in “**LOG_OF_LOGS Report**” on page 416.

Using the **LOG_OF_LOGS** Keyword

Use the required **LOG_OF_LOGS** keyword to identify the z/OS log of logs as follows:

- On the **REGISTER**, **ADD**, and **UPDATE** **LOG_OF_LOGS** command, set the value to the fully qualified log stream name.

- On the **DELETE** **LOG_OF_LOGS** command, set the value to the log stream name. You can use masking (wildcard) characters in the name. You can include multiple names; use a separator between the names.

Using the **DELETE** Keyword

Use the optional **DELETE** keyword on the **REGISTER LOG_OF_LOGS** command to control whether RUV deletes the data that has been read from the log stream. Set one of the following values. The default value is NO.

**NO** Leave the data intact. CICS tie-up records are written to the log of logs each time a CICS VSAM file is opened or closed; consequently, the log of logs slowly grows. You may want to run a delete job once a month or quarterly to clean out old information.

**YES** Delete the data from the log stream.
Using the DAYS Keyword

Use the optional DAYS keyword on the REGISTER LOG_OF_LOGS command to limit the number of days of data to be scanned; for example, DAYS(7) causes RUV to scan the last week of activity. Set the value to the number (1 to 999) of days.

The DAYS keyword on the SET DEFAULT command does not apply to the log of logs information. For more information, see “Using the DAYS Keyword” on page 131.

Using the START_TIME Keyword

Use the optional START_TIME keyword on the REGISTER LOG_OF_LOGS command to set the starting time for the scan. For more information, see “Using the START_TIME Keyword” on page 111.

Using Optional Keywords

You can use the following optional keywords on the ADD LOG_OF_LOGS and UPDATE LOG_OF_LOGS commands:

- COMMENT (see “Using the COMMENT Keyword” on page 107)
- STATUS (see “Using the STATUS Keyword” on page 112)

Coding ADD LOG_OF_LOGS Statements

To code an ADD LOG_OF_LOGS statement, use the following syntax:

```
ADD LOG_OF_LOGS(/sn26)
   { COMMENT(comment_fields) }
   { STATUS(ACTIVE | INACTIVE) }
;
```
Coding DELETE LOG_OF_LOGS Statements

To code a DELETE LOG_OF_LOGS statement, use the following syntax:

```
DELETE LOG_OF_LOGS(lsn26, *, ...) ;
```

Coding REGISTER LOG_OF_LOGS Statements

To code a REGISTER LOG_OF_LOGS statement, use the following syntax:

```
REGISTER LOG_OF_LOGS(lsn26, *, ...) {
  DAYS(nn) | START_TIME(yyyyjjjjhhmmssst) 
  DELETE (YES | NO) 
} ;
```

Coding UPDATE LOG_OF_LOGS Statements

To code an UPDATE LOG_OF_LOGS statement, use the following syntax:

```
UPDATE LOG_OF_LOGS(lsn26) {
  COMMENT(comment_fields) 
  STATUS(ACTIVE | INACTIVE) 
} ;
```
Working with Archives

This chapter discusses how to work with archives. The following topics are included:

Overview of Archive Management .......................................................... 247
Accumulating Archive Data Sets .............................................................. 248
  Separating Archives ............................................................................. 248
  Merging Archives and Combining Data ............................................... 248
  Retaining and Purging Input Archives ................................................. 249
  Setting Up the Accumulation Job ......................................................... 249
  Using the ARCHIVE ACCUM Command ............................................ 250
  Using the INVENTORY_ONLY Keyword ............................................. 251
  Using the ARCHIVE_IN Keyword ..................................................... 251
  Using the SCRATCH Keyword ........................................................... 252
  Using the ACCUM Keyword ............................................................. 254
  Using the CONSOLIDATED Keyword ................................................ 254
  Using the VSAM_FILE Keyword ......................................................... 254
  Using the VSAM_GROUP Keyword ..................................................... 254
  Using the UNSELECTED_VSAM_FILES Keyword ............................. 254
  Using the RETAIN_IMAGE Keyword ............................................... 255
  Using the LAST_IMAGE Keyword .................................................... 255
  Using the ARCHIVE_OUT Keyword ................................................ 256
  Suggested Usage and Example 1 ....................................................... 256
  Suggested Usage and Example 2 ....................................................... 259
Copying Archive Data Sets ................................................................. 260
  Using the ARCHIVE ARCHIVE_IN Command .................................. 260
  Using the ARCHIVE_IN Keyword .................................................... 260
  Using the ARCHIVE_OUT Keyword ................................................. 261
  Coding ARCHIVE ARCHIVE_IN Statements .................................... 261
Registering Archive Data Sets ............................................................. 262
  Using the REGISTER ARCHIVE_IN Command ................................ 262
  Using the ARCHIVE_IN Keyword .................................................... 262
  Using Optional Keywords ............................................................. 262
  Coding REGISTER ARCHIVE_IN Statements .................................... 263
Updating Archive Records ................................................................. 264
   Using the UPDATE ARCHIVE_FILE Command ................................. 264
   Using the ARCHIVE_FILE Keyword .............................................. 265
   Using the FILTER_BY Keyword .................................................... 265
   Using the REGISTRATION_TIME Keyword ...................................... 265
   Using Optional Keywords .......................................................... 266
   Coding UPDATE ARCHIVE_FILE Statements .................................... 266
Deleting Archive Records ............................................................... 268
   Using the DELETE ARCHIVE_FILE Command .................................. 268
   Using the ARCHIVE_FILE Keyword .............................................. 268
   Using the FILTER_BY Keyword .................................................... 268
   Using the SCRATCH Keyword ........................................................ 269
   Coding DELETE ARCHIVE_FILE Statements .................................... 269
Reporting Archive Information in the Repository ................................. 270
   Using the REPORT ARCHIVE_FILE Command ................................ 270
   Using the ARCHIVE_FILE Keyword .............................................. 270
   Using the FILTER_BY Keyword .................................................... 271
   Coding REPORT ARCHIVE_FILE Statements .................................... 271
Archiving RPCV Journals ................................................................. 272
   Using the ARCHIVE RPCV_IN Command ...................................... 272
   Using the RPCV_IN Keyword ...................................................... 272
   Using the COMPLETE_COPY Keyword ......................................... 272
   Using Other Keywords .................................................................. 273
   Coding ARCHIVE RPCV_IN Statements ........................................ 273
Printing Archive Data ......................................................................... 274
   Using the PRINT ARCHIVE_IN Command ...................................... 274
   Using the ARCHIVE_IN Keyword ................................................. 274
   Using the COUNT Keyword .......................................................... 274
   Using the SELECTION_EXIT keyword ............................................ 275
   Coding PRINT ARCHIVE_IN Statements ........................................ 276
Overview of Archive Management

The collection and management of archives is important to providing for a fast, accurate recovery, especially if you are performing the recovery off-site. Archive management has multiple goals:

- **reduction of number of archives**
  
  The fewer archives that you must handle, the less chance of error, especially when you are moving archives off-site.

- **efficient resource use**
  
  A smaller number of archives consume less DASD or tape space. Off-site resource storage usually commands a premium price.

- **speed of recovery**
  
  Most users want to minimize the amount of time for a recovery.

RUU provides the same journal record capability for batch VSAM processing that CICS provides for online processing. Journal records are images of file records before or after a transaction update. These journal records include all updates, additions, and deletions of records within a data set. RUV uses the changes reflected in the journal records to reconstruct the data set if a full recovery is needed.

These changes are recorded in RUV archives. RUV potentially produces many archive files. It produces an archive file for each CICS journal extent switch, for each log stream archive, and for each step in a batch job that has recoverable VSAM files. (See note on page 36 concerning journal support for unsupported CICS releases.)

Internally, RUV tracks date and time information with a 64-bit timestamp. You might occasionally see a zero date. This is normal for an empty archive. An empty archive does not contain any recovery data. The zero date eliminates consideration of the archive for recovery purposes. Empty archives are deleted by the REPOSITORY PURGE command. You might also see dates with the year 2042. This date typically indicates that an end time has not been set, and it is interpreted as up to current time.

**NOTE**

Before you archive a log stream, it is recommended that you issue the CEMT SET JOU(DFHJnn) FLUSH command. See the IBM CICS Supplied Transactions manual for more information.
Accumulating Archive Data Sets

The RUV accumulation function can separate archive data for a file, or data for a group of files, into separate archives. It can also merge the archive data for a file, or a group of files, into a single archive.

It also provides last-image accumulation of forward recovery data. RUV sorts the data, discards all the after-image records except the most current records, and writes those records to a last image archive.

When performing a recovery, RUV selects all of the active archives for processing, even if there are copies of the same archive file. It can retain or delete the input archives.

With careful planning, the accumulation function is a powerful tool for improving recovery speed and minimizing storage requirements.

Separating Archives

If you have hundreds of files on one archive, but you need to recover one of these files quickly, you can use the accumulation function (or multiple executions of the function) to specify that file alone for recovery.

Merging Archives and Combining Data

Periodic accumulation of archives helps you to optimize DASD and tape resources. This process usually involves moving recovery data from DASD to less-expensive media, such as tape. You should perform the accumulation process periodically—hourly, daily, or weekly, depending on your needs. After the recovery data has been copied from many small archives to one or more larger archives, RUV can delete the input archives.

The accumulation process also allows you to combine recovery data. You should place files or groups that are normally recovered together on the same output archive. This arrangement typically speeds up a recovery because unnecessary data does not have to be passed during the recovery process.

RUV groups contain selected VSAM files under a single group name. Organization of VSAM backup or recovery records can be streamlined and made more efficient with the group feature. See Chapter 15, “Using VSAM Groups.”
Retaining and Purging Input Archives

The consolidated accumulation process has many built-in safeguards. Unless instructed otherwise, RUV retains archives that are used for input to the accumulation process (they are made inactive, but not deleted). RUV can be instructed to purge the input archives from the RUV repository and scratch them from the z/OS catalog after the accumulation process completes; however, RUV does not purge or scratch them until all needed recovery information in them has been successfully written to consolidated accumulation archives. This feature allows you to use an archive that contains active records (records that have not already been accumulated) as input to several accumulation jobs. Records on a journal that have been consolidated once will not be selected again unless the archive is reregistered. RUV allows you to create multiple copies of output archives during the accumulation process.

NOTE
Be cautious about consolidating too many VSAM files on a single archive. A poorly planned consolidated archive reduces the speed of recovery if it reduces the number of archives that can be read in parallel.

Tape MOD operations cannot be performed.

Setting Up the Accumulation Job

In general, the accumulation process is fairly simple. You specify the VSAM data set names for RUV to select and specify where to write the archive records. RUV automatically selects for accumulation the active archives that contain data for the specified VSAM files and that are not accumulations. Figure 45 on page 250 illustrates this process.
Using the ARCHIVE ACCUM Command

A warning message is issued when the archives of the VSAM file are older than the backup file.

Using the ARCHIVE ACCUM Command

To control the archive accumulation process with the ARCHIVE ACCUM statement, use the following syntax.

```plaintext
ARCHIVE
   { INVENTORY_ONLY(NO | YES) }
   { ARCHIVE_IN(ddn_or_dsn, *, ...) }
   { SCRATCH(NO | YES | FORCE) }
   ACCUM(
      { CONSOLIDATED }
      { VSAM_FILE(ddn8_or_dsn44, *, ...) }
      { UNSELECTED_VSAM_FILES }
      { RETAIN_IMAGE(BOTH | LOG | JOURNAL) }
      { LAST_IMAGE(name, *, ...) }
      { VSAM_FILE(name, *, ...) }
      { VSAM_GROUP(name, *, ...) }
      { RETAIN_IMAGE(JOURNAL) }
      ARCHIVE_OUT(ddn
          { LOCATION(locname) }
          { COMMENT(comment_fields) }
          { STATUS(ACTIVE | INACTIVE) }
          { MUST_COMPLETE (YES | NO)}
      )
   )
   ACCUM( ... )
; 
```
Using the INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the archive command to show a listing of VSAM Group files that will be archived, and the name of each archive to be created by this job. If you specify INVENTORY_ONLY(YES), archive processing will not occur but a report will be produced.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

**YES** Prevent archive processing and issue a preliminary report on files that will be archived.

**NO** Perform archive processing.

Using the ARCHIVE_IN Keyword

You can use the optional ARCHIVE_IN keyword on the ARCHIVE ACCUM command to identify the archives to use as input. If you specify this keyword, RUV uses only the specified archives for input. If you omit this keyword, RUV automatically selects the necessary input archives based on the names of the VSAM files that are provided in the ACCUM data sets. In either case, RUV processes active archives only.

Set the ARCHIVE_IN(ddname) keyword value to the ddname of the input archive data sets in the JCL, or set it to the data set names to have RUV dynamically allocate the data sets. You can use masking (wildcard) characters in the data set name, and you can provide multiple values. Separate the values with a comma and a blank space.

If you specify the input archives with the ARCHIVE_IN keyword, you are responsible for providing enough sort work space for sorting the number and size of input archive records. If RUV selects the archives (you omit the ARCHIVE_IN keyword), RUV automatically requests the appropriate amount of sort work space.

RUV can accumulate unregistered archives only if you specify them with the ARCHIVE_IN keyword; if the archive is not registered, the job returns a condition code of 4.

**NOTE**
The accumulation process does not automatically select input archives that were previously built by accumulation for consolidated, but it does do so for last image. If you want to use an accumulation archive as an input to a consolidated accumulation, you must specify it with the ARCHIVE_IN keyword.
Using the SCRATCH Keyword

You can use the optional SCRATCH keyword to control the disposition of the input archives after the accumulation process has completed. Specify one of the following values:

**NO**  Leave the input archives cataloged and registered in the RUV repository, but mark selected archive data as inactive. This value is the default.

**YES**  Scratch the input archive from the z/OS catalog and delete the archive record from the RUV repository if all files on the input archive have been marked as inactive.

**FORCE**  Scratch the input archive from the z/OS catalog and delete the archive record from the RUV repository after the data is read, without checking the status for the archive that was processed.

---

**WARNING**  Specify SCRATCH(FORCE) only with extreme caution.

Using the SCRATCH Keyword with Mixed Archives

Depending on the CICS file definition settings and specific RUV command options, it is possible to accumulate archives with the SCRATCH(YES) keyword and not delete all of the input archives. RUV is designed to retain an archive if it contains a record that has not been read and acted on.

The following examples illustrate how CICS file definition settings and the RUV keywords affect the action of the ACCUM and SCRATCH keywords.

**Example 1**  When the RECOVERY(ALL) and AUTOJOURNAL options are specified, the RUV archive contains both forward and backout records. The use of the RETAIN_IMAGE() and SCRATCH(YES) keywords deletes the archive files from the repository and scratches the input archive from the system catalog after the accumulation process.

**Example 2**  When the RECOVERY(ALL) and NO AUTOJOURNAL options are specified, the RUV archive contains forward records only. If the RETAIN_IMAGE(LOG) and SCRATCH(YES) keywords are specified, RUV does not delete or scratch the input archive because RUV is instructed to read and process the log records.
Example 3  When the RECOVERY(NONE) and AUTOJOURNAL options are specified, the RUV archive contains backout records only. If the RETAIN_IMAGE(JOURNAL) and SCRATCH(YES) keywords are specified, RUV does not delete or scratch the input archive because RUV is instructed to read and process the journal records.

It is possible to resolve these problems by reviewing all your JCL jobs and changing the CICS file definition settings or by specifying the RETAIN_IMAGE(BOTH) keyword in the accumulation job. However, this action might produce larger archives and impact system performance.

Coding the SCRATCH Keyword for Mixed Archives

The preferred method for accumulating data from archives with a mix of before-image and after-image data is to collect each type of data in a separate file and then delete the unwanted data when the job is done. The file containing the unwanted data should be a virtual (dummy) file.

Code the accumulation of archive records with a mix of before-image and after-image data as shown in the following example. This example is designed to retain the data to perform a forward recovery. If you want to perform a backout recovery, swap the RETAIN_IMAGE keywords so the user log before-images are retained.

```
//NOARCH DD DUMMY
//SYSIN DD *
SET SUBSYSTEM(ssid);
SET REPORT(DETAIL);
ARCHIVE
  SCRATCH(YES)
  VSAM_FILE(*)
    ACCUM(
      RETAIN_IMAGE(JOURNAL)
      CONSOLIDATED
      ARCHIVE_OUT(ARCH
      COMMENT("ALL AP PRODUCTION FILES")
      MUST_COMPLETE(YES)))
    ACCUM(
      RETAIN_IMAGE(LOG)
      CONSOLIDATED
      ARCHIVE_OUT(NOARCH) STATUS(INACTIVE));
    COMMENT("RECORDS TO DELETE")
    DELETE ARCHIVE_FILE(NULLFILE);
```

The processing needed to collect the unwanted data in the dummy file will have a negligible impact on speed of the accumulation. A report is generated on the data going to the dummy file and the dummy file is registered into the repository.

You can use the DELETE command to remove the archive immediately, or you can allow the PURGE REPOSITORY command to remove it later.
Using the ACCUM Keyword

Use the required ACCUM keyword to tell RUV that you want to perform the Accumulation function.

Using the CONSOLIDATED Keyword

You can use the optional CONSOLIDATED keyword to specify that RUV should write a consolidated archive.

Using the VSAM_FILE Keyword

You can use the optional VSAM_FILE keyword to identify the VSAM data set names to be selected for a set of archive outputs. You can select the same VSAM file for more than one archive group. The VSAM_FILE keyword is mutually exclusive with the UNSELECTED_VSAM_FILES keyword.

**NOTE**

If the VSAM_FILE data set names are not found in the repository, or if the base of a specified path is not found, the accumulation process halts with a condition code of 8.

Using the VSAM_GROUP Keyword

Use the optional VSAM_GROUP(name32) keyword to use a 32-character VSAM group record in the repository. (See Chapter 15, "Using VSAM Groups" for further explanation of groups.)

Using the UNSELECTED_VSAM_FILES Keyword

You can use the optional UNSELECTED_VSAM_FILES keyword to cause RUV to write to the associated ARCHIVE_OUT file or files any archive data for files that were not selected to any other accumulation group. This keyword does not cause any additional automatic ARCHIVE_IN selection.
Using the RETAIN_IMAGE Keyword

You can use the optional RETAIN_IMAGE keyword to control which types of archive images to copy. Specify one of the following values:

**BOTH**
Select before-images and after-images. This value is the default.

**LOG**
Select before-images to allow backout recovery.

**JOURNAL**
Select after-images to allow forward recovery. When using the LAST_IMAGE keyword, JOURNAL is the default and the only selection available.

Using the LAST_IMAGE Keyword

You can use the optional LAST_IMAGE keyword to accumulate last image records of forward recovery data only.

To perform LAST_IMAGE accumulation, RUV requires the following input data sets to be registered in the repository:

- at least one archive
- at least one backup of each VSAM file for which changes are recorded in the archive

RUV processes only one of the most current archives if copies of that archive exist. The duplicate copies of that archive are not allocated or processed, which improves the performance of accumulation processing.

When you include the LAST_IMAGE keyword, RUV processes all archive copies that are earlier than the latest archive and were taken after the backups which are included in the accumulation processing.

---

**NOTE**
Using the UNSELECTED_VSAM_FILES keyword by itself causes RUV to accumulate all VSAM data sets from the ARCHIVE_IN data set.
Using the ARCHIVE_OUT Keyword

Use the required ARCHIVE_OUT keyword on the ARCHIVE ACCUM command to provide the ddnames of the output archives. RUV can create more than one copy of the output archives. RUV registers all output archives in the repository. The new archives carry recovery data only. RUV accumulation drops job-related information.

**NOTE**

You cannot specify the same ARCHIVE_OUT data set in more than one ACCUM group.

You can use the following optional keywords with the ARCHIVE_OUT keyword on the ARCHIVE ACCUM command:

- COMMENT (see “Using the COMMENT Keyword” on page 107)
- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- MUST_COMPLETE (see “Using the MUST_COMPLETE Keyword” on page 110)

Code these keywords and their values after the ddname value but before the closing parenthesis for the ARCHIVE_OUT value. Only one archive-per-accumulation-set copy should have an active status.

**NOTE**

It is recommended that you do not use ARCHIVE_OUT to refer to an archive that is registered in the RUV repository. If you reuse ARCHIVE_OUT data sets when they are still registered in the repository, accumulation failures may result from an attempt to use the same archive as input and output.

Suggested Usage and Example 1

This example shows the use of accumulation to free resources, to speed recovery of selected data sets, and to simplify processes.

You could have all general ledger files written to accumulation one, all inventory files to accumulation two, general ledger and inventory files to accumulation three, and all other data sets to accumulation four. Then accumulation one and two could be retained on site for fast local recovery, while accumulation three is sent to off-site storage. All accumulations will save only forward recovery data (JOURNAL).
Accumulations one, two, and four are made active, while three is made inactive. Since you are making two copies of the critical data, you want to scratch the input archives. Finally, if there is a problem with accumulation four, let the job continue, but halt if any other accumulation has a problem.

The following naming conventions are in effect in this example:

- all general ledger data sets start with GNL*
- all inventory files data sets start with INVTORY.*
- the accumulated output general ledger is put in a GDG named GNLACCM.DATA(+1)
- the accumulated output inventory files is put in a GDG named INVTORY.ACCM.DATA(+1)
- the accumulated output of both files is put in a file named GNLINVNTY.ACCM.DATA.AG072009
- the accumulated output of all other data sets is put in a file named OTHER.ACCM.AG072009

Figure 46 shows an example of suggested ACCUM usage to free resources, speed up recovery, and simplify processes.

**Figure 46  Usage Example of ACCUM Process**

![Diagram showing the usage example of ACCUM process]

Figure 47 shows the JCL and control statements to free resources, speed up recovery, and simplify processes.

**Figure 47  Accumulation Example 1 (Part 1 of 2)**

```
//ACCMCDATA JOB . . .
//ACCM EXEC PGM=RUZSMO
//SYSPRINT DD SYSOUT=*  
```
Suggested Usage and Example 1

Figure 47  Accumulation Example 1 (Part 2 of 2)

```plaintext
//GNLOUT DD DSN=GNLACCM.DATA(+1),
// DISP=(NEW,CATLG),UNIT=SYSDA,
// SPACE=(CYL,(10,10),RLSE),
// DCB=GNL.MODDSCB,FREE=CLOSE
// *
//INVTOUT DD DSN=INVTORY.ACCM.DATA(+1),
// DISP=(NEW,CATLG),UNIT=SYSDA,
// SPACE=(CYL,(5,3),RLSE),
// DCB=INVTORY.MODDSCB,FREE=CLOSE
// *
//GNLINVTO DD DSN=GNLINVTY.ACCM.DATA.AG072009,
// DISP=(NEW,CATLG),UNIT=TAPE,FREE=CLOSE
// *
//OTHEROUT DD DSN=OTHER.ACCM.AG072009,
// DISP=(NEW,CATLG),UNIT=TAPE,FREE=CLOSE
// *
//SYSIN    DD   *
ARCHIVE
  SCRATCH(YES)
  ACCUM(
    VSAM_FILE(GNL*)
    RETAIN_IMAGE(JOURNAL)
    CONSOLIDATED
    ARCHIVE_OUT(GNLOUT
      COMMENT("GENERAL LEDGER DATA - 08/07/2009")
        MUST_COMPLETE(YES)))
  ACCUM(
    VSAM_FILE(INVTORY.*)
    RETAIN_IMAGE(JOURNAL)
    CONSOLIDATED
    ARCHIVE_OUT(INVTOUT
      COMMENT("INVENTORY DATA - 08/07/2009")
        MUST_COMPLETE(YES)))
  ACCUM(
    VSAM_FILE(INVTORY.*, GNL*)
    VSAM_GROUP(INVENTORY)
    RETAIN_IMAGE(JOURNAL)
    CONSOLIDATED
    ARCHIVE_OUT(GNLINVTO
      LOCATION(OFFSITE)
      COMMENT("INVENTORY DATA - 08/07/2009" 
        "GENERAL LEDGER DATA - 08/07/2009")
        STATUS(INACTIVE)
        MUST_COMPLETE(YES)))
  ACCUM(
    UNSELECTED_VSAM_FILES
    RETAIN_IMAGE(JOURNAL)
    CONSOLIDATED
    ARCHIVE_OUT(OTHEROUT
      LOCATION(LOCAL)
      COMMENT("OTHER RECOVERY DATA - 08/07/2009")
        STATUS(ACTIVE)
        MUST_COMPLETE(NO)));
```
This example demonstrates the use of accumulation to free resources and to simplify your processes.

You have been creating daily accumulations, and now want to roll them together into a weekly accumulation. Since you still have the off-site copy for emergencies, you will have the accumulation process scratch the inputs. Since accumulations cannot be automatically selected in consolidated, they must be input as ARCHIVE_IN. All data from the inputs are to be placed on the output.

The following naming conventions are in effect:

- the input accumulations are in GDGs named GNLACCM.DATA.G00Vnnn and in GDGs named INVTORY.ACCM.DATA.G00Vnnn
- the accumulated output is put on a tape file named GNLINVVTY.WKLYACCM.DATA.AG132009

Figure 48 shows the JCL and control statements to accomplish this task.

**Figure 48  Accumulation Example 2**

```plaintext
//ACCMDATA JOB . . .
//ACCM  EXEC PGM=RUVZSM0
//SYSPRINT DD SYSOUT=*  
//*  
//WKACCM DD DSN=GNLINVTY.WKLYACCM.DATA.AG132009, 
//   DISP=(NEW,CATLG),UNIT=TAPE,FREE=CLOSE 
//*  
//SYSIN DD *  
ARCHIVE  
ARCHIVE_IN(GNLACCM.DATA.*) 
INVTORY.ACCM.DATA.* 
SCRATCH(YES) 
ACCUM( 
   VSAM_FILE(*) 
   RETAIN_IMAGE(JOURNAL) 
   CONSOLIDATED 
   ARCHIVE_OUT(WKACCM 
      COMMENT("WEEKLY DATA - 08/13/2009") 
      MUST_COMPLETE(YES) 
   ) 
)  
;  
```
Copying Archive Data Sets

You can copy or move an archive from one device to another. RUV registers the new archives in the repository.

**NOTE**

Do not concatenate input data sets. If you want to combine archives, use the ARCHIVE ACCUM command. Combining offsite storage functions with use of the LOCATION keyword can simplify disaster recovery preparations.

Using the ARCHIVE ARCHIVE_IN Command

To copy or move an archive, use the ARCHIVE ARCHIVE_IN command, as shown in the following example.

```
ARCHIVE ARCHIVE_IN(GNL.ARCVDASD)
   SCRATCH(YES)
   ARCHIVE_OUT(ARCVTAPE
      COMMENT('GENERAL LEDGER LOCAL ARCHIVE')
      LOCATION(LOCAL)
      STATUS(ACTIVE)
      MUST_COMPLETE(YES)
    )
   ARCHIVE_OUT(ARCVTAP2
      COMMENT('GENERAL LEDGER REMOTE ARCHIVE')
      LOCATION(REMOTE)
      STATUS(INACTIVE)
      MUST_COMPLETE(NO)
    )
;
```

Using the ARCHIVE_IN Keyword

Use the required ARCHIVE_IN keyword on the ARCHIVE ARCHIVE_IN command to specify the data set name of the archive to copy or move. Set the value to the ddname of the input archive data set in the JCL, or set it to the fully qualified data set name to have RUV dynamically allocate the data set.
Using the ARCHIVE_OUT Keyword

Use the required ARCHIVE_OUT keyword on the ARCHIVE ARCHIVE_IN command to specify the new data set name of the copied or moved archive. Set the value to the ddname of the output archive data set in the JCL.

You can specify as many ARCHIVE_OUT keywords as the number of copies you need, limited by the available system resources (the storage buffers for the outputs that are created).

You can code the following optional keywords with the ARCHIVE_OUT keyword on the ARCHIVE ARCHIVE_IN command:

- COMMENT (see “Using the COMMENT Keyword” on page 107)
- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- MUST_COMPLETE (see “Using the MUST_COMPLETE Keyword” on page 110)

Code these keywords and their values after the ddname value but before the closing parenthesis for the ARCHIVE_OUT value. Only one archive copy should have an active status.

Coding ARCHIVE ARCHIVE_IN Statements

To code an ARCHIVE ARCHIVE_IN statement, use the following syntax:

```
ARCHIVE ARCHIVE_IN(ddn8_or_dsn44)
  { SCRATCH(NO | YES) }
ARCHIVE_OUT(ddn8
  { LOCATION(locname8) }
  { COMMENT('comment fields') }
  { STATUS(ACTIVE | INACTIVE) }
  { MUST_COMPLETE(YES | NO) }
)
  { ARCHIVE_OUT(ddn8, ...) }
;
```

For clarity, you can also use the following unit-of work-syntax to copy archives that contain UOW data:

```
ARCHIVE
  UNIT_OF_WORK_IN(ddn8_or_dsn44)
  UNIT_OF_WORK_OUT(ddn8) ;
```
Registering Archive Data Sets

Normally, RUV registers archive data sets automatically; no action is required. You can register an archive manually if necessary, for example, if an archive data set has been deleted accidentally or you need to add it to a different repository.

Using the REGISTER ARCHIVE_IN Command

To register an archive, use the REGISTER ARCHIVE_IN command, as shown in the following example.

```
REGISTER ARCHIVE_IN(GNL.ARCVLEDG)
  LOCATION(LOCAL)
  COMMENT('GENERAL LEDGER LOCAL ARCHIVE')
  STATUS(ACTIVE)
```

Using the ARCHIVE_IN Keyword

Use the required ARCHIVE_IN keyword on the REGISTER ARCHIVE_IN command to identify the archive to register. Set the value to the ddname of the input archive data set in the JCL, or set it to the fully qualified data set name.

Using Optional Keywords

You can use the following optional keywords on the REGISTER ARCHIVE_IN command:

- COMMENT (see “Using the COMMENT Keyword” on page 107)
- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
Coding REGISTER ARCHIVE_IN Statements

To code a REGISTER ARCHIVE_IN statement, use the following syntax:

```sql
REGISTER ARCHIVE_IN(ddn8_or_dsn44)
  { LOCATION(locname8) }
  { COMMENT('comment fields') }  
  { STATUS(ACTIVE | INACTIVE) } 
;  
```
You can change repository information for an archive if necessary, for example, if the status should be active so that RUV will select the archive for recovery.

**NOTE**
If you alter the status of any archive, any subsequent archive and recovery processing could be adversely affected. Use this command on your test data files when recovery and archive processing are not a concern.

### Using the UPDATE ARCHIVE_FILE Command

To update an archive record, use the UPDATE ARCHIVE_FILE command, as shown in the following example:

```sql
UPDATE ARCHIVE_FILE(GNL*)
  FILTER_BY(
    LOCATION(REMOTE)
    STATUS(INACTIVE)
    START_TIME(20092580000000)
    STOP_TIME(20092590000000)
  )
  LOCATION(DRSITE)
  COMMENT('DISASTER RECOVERY')
  STATUS(ACTIVE)
;
```

You can also update a specific archive record based on a registration time in the repository, as shown in the following example:

```sql
UPDATE ARCHIVE_FILE(GNL.ARCVLEDG)
  LOCATION(REMOTE)
  COMMENT('DISASTER RECOVERY')
  STATUS(INACTIVE)
  REGISTRATION_TIME(20092920518450)
;
```
Using the ARCHIVE_FILE Keyword

Use the required ARCHIVE_FILE keyword on the UPDATE ARCHIVE_FILE command to identify the archive data set. Specify one of the following values:

- the ddname of the input archive data set in the JCL
- a single fully qualified data set name if you use the REGISTRATION_TIME keyword

The data set name cannot contain wildcard characters, and you cannot specify multiple data set names.

- one or more fully qualified data set names, data set name masks, or both, if you do not use the REGISTRATION_TIME keyword

You can use wildcard characters (* and ?) and specify multiple values.

Using the FILTER_BY Keyword

Use the optional FILTER_BY keyword to filter the selection of archive records to update. You can filter the selection of archive records by location, status, start time, and stop time. For more information, see “Using the FILTER_BY Keyword” on page 108.

If you specify the FILTER_BY keyword, do not specify the REGISTRATION_TIME keyword.

Using the REGISTRATION_TIME Keyword

Use the optional REGISTRATION_TIME keyword to select a specific registered archive record to be modified in the repository. This keyword is required for automatic deactivation of journals after a backup that has been automatically submitted has been executed.

If you include the <RUV_JOURNAL_DEACTIVATE> variable in a rule set that is defined to perform a backout recovery for a job step that terminates abnormally, in the backout step RUV generates the following command for each log or journal that was produced by the job step that terminates abnormally:

```
UPDATE ARCHIVE_FILE('dsn44')
    REGISTRATION_TIME('Xhhhhhhhhhhhhhhhh')
    STATUS(INACTIVE);
```
Using Optional Keywords

The value of the REGISTRATION_TIME keyword is a timestamp. You can provide the timestamp in the format \texttt{yyyyjjjhhmmssst} or as the hexadecimal value \texttt{(Xhhhhhhhhhhhhhhhh)} that RUV reports.

RUV updates the record that has the first matching data set name with a registration time that is greater than or equal to the value of the REGISTRATION_TIME keyword. RUV ignores all other archives. The REGISTRATION_TIME keyword is valid for a single, specific data set name or ddname; it is invalid with a list of data set names or ddnames and with wildcard selection.

If you specify the REGISTRATION_TIME keyword, do not specify the FILTER_BY keyword.

Using Optional Keywords

You can use the following optional keywords on the UPDATE ARCHIVE_FILE command:

- COMMENT (see “Using the COMMENT Keyword” on page 107)
- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)

Coding UPDATE ARCHIVE_FILE Statements

To code an UPDATE ARCHIVE_FILE statement with the FILTER_BY keyword, use the following syntax:

```sql
UPDATE ARCHIVE_FILE(ddn8_or_dsn44, *, ...)
  { FILTER_BY(
      { LOCATION(location_name8,*, ...) }
      { STATUS(ACTIVE | INACTIVE) }
      { START_TIME(yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh) }
      { STOP_TIME(yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh) }
  )
  { LOCATION(location_name8) }
  { COMMENT('comment fields') }
  { STATUS(ACTIVE | INACTIVE) }

;```
To code an UPDATE ARCHIVE_FILE statement with the REGISTRATION_TIME keyword, use the following syntax:

**Figure 49 UPDATE ARCHIVE_FILE statement with REGISTRATION_TIME keyword**

```plaintext
UPDATE ARCHIVE_FILE(ddn8_or_dsn44)
   { REGISTRATION_TIME(yyyyjjjhmmssst | Xhhhhhhhhhhhhhh) }
   { LOCATION(location_name8) }
   { COMMENT('comment fields') }
   { STATUS(ACTIVE | INACTIVE) }

;  
```
Deleting Archive Records

Normally, RUV deletes archive records from the repository during the scheduled purge cycle. You can delete specific archive records.

**WARNING**

It is recommended that you use this command only to discard test data or data that was created erroneously. This command might cause data integrity issues when used against active archives.

Using the **DELETE ARCHIVE_FILE** Command

To delete an archive record from the repository, use the **DELETE ARCHIVE_FILE** command, as shown in the following example:

```
DELETE ARCHIVE_FILE(GNL.ARCV*)
FILTER_BY(
  LOCATION(REMOTE)
  STATUS(INACTIVE)
  START_TIME(20092500000000)
  STOP_TIME(20092520020000)
)
SCRATCH(YES)
;
```

Using the **ARCHIVE_FILE** Keyword

Use the required **ARCHIVE_FILE** keyword on the **DELETE ARCHIVE_FILE** command to identify the archive record (or records) to delete from the repository. Set the value to the data set name of the archive data set, or set it to the ddname of the archive data set in the JCL. You can use masking (wildcard) characters. You can provide multiple values; use a separator between the values.

Using the **FILTER_BY** Keyword

Use the optional **FILTER_BY** keyword to filter the selection of archive records to be deleted. You can filter the selection of archives by location, status, start time, and stop time. For more information, see "Using the **FILTER_BY** Keyword" on page 108.
Using the SCRATCH Keyword

You can use the optional SCRATCH keyword to control the handling of input archives after the accumulation process has completed. For more information, see “Using the SCRATCH Keyword” on page 252.

Coding DELETE ARCHIVE_FILE Statements

To code a DELETE ARCHIVE_FILE statement, use the following syntax:

```plaintext
DELETE ARCHIVE_FILE(ddn8_or_dsn44)
   { FILTER_BY(
      { LOCATION(location_name8,.*, ...) }
      { STATUS(ACTIVE | INACTIVE) }
      { START_TIME(yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhh) }
      { STOP_TIME(yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhh) }
   )
   { SCRATCH(NO | YES) }
```


Reporting Archive Information in the Repository

RUV can provide a report about the archive records that are stored in the repository. You can use this type of report to determine the results of a command before the actions are performed. For example, you can run a REPORT ARCHIVE_FILE command to see which archive records would be deleted if you changed the command to DELETE ARCHIVE_FILE.

For an example of the report that is produced by the REPORT ARCHIVE_FILE command, see “ARCHIVE_FILE Report” on page 405.

Using the REPORT ARCHIVE_FILE Command

To report information about the archive records in the repository, use the REPORT ARCHIVE_FILE command, as shown in the following example:

```REPORT ARCHIVE_FILE(GNL.LEDGBK*, GNL.PAYRBK*)
FILTER_BY(
  START_TIME(20092460000000)
)
```

In this example, the START_TIME keyword specifies that you want to select archive files containing records that were created since the specified time.

Using the ARCHIVE_FILE Keyword

Use the required ARCHIVE_FILE keyword on the REPORT ARCHIVE_FILE command to identify the archive records to select. The value can be the specific ddname of the archive file in the JCL, or it can be the data set name of the archive file; a data set name value can contain masking (wildcard) characters. You can code multiple values for this keyword.

The REPORT ARCHIVE_FILE command does not read the archive data set. It uses the name you supply only to identify the archive record in the repository. However, if you specify the ddname, z/OS allocation of the data set occurs. It is recommended that you use the data set name instead of the ddname so that z/OS allocation does not occur.
Using the FILTER_BY Keyword

Use the optional FILTER_BY keyword to further refine or filter the selection of archive records to be reported. You can filter the selection of archive records by location, status, start time, and stop time. For more information, see “Using the FILTER_BY Keyword” on page 108.

Coding REPORT ARCHIVE_FILE Statements

To code a REPORT ARCHIVE_FILE statement, use the following syntax:

```
REPORT ARCHIVE_FILE(ddn_or_dsn, *, ..)
  { FILTER_BY(
    { LOCATION(location_name8, *, ...) }
    { STATUS(ACTIVE | INACTIVE) }
    { START_TIME/yyyyjjjjhhmmssst | Xhhhhhhhhhhhhhhhh }
    { STOP_TIME/yyyyjjjjhhmmssst | Xhhhhhhhhhhhhhhhh }
  )
};
```
Archiving RPCV Journals

You can convert the journals that were created by the BMC Software RECOVERY PLUS for CICS/VSAM (RPCV) product into a format that can be used with RUV. Any RPCV journals that you want to use with RUV must be processed by the ARCHIVE RPCV_IN command so that they are registered for recovery processing.

Using the ARCHIVE RPCV_IN Command

Use the ARCHIVE RPCV_IN command, as shown in the following example, to convert an RPCV journal to RUV format and register it in the repository.

```
ARCHIVE RPCV_IN(RPCVJOUR)
  COMPLETE COPY
  ARCHIVE_OUT(RUVARCH1
    LOCATION(LOCAL)
    COMMENT('LOCAL RPCV JOURNAL / RUV ARCHIVE')
    STATUS(ACTIVE)
    MUST_COMPLETE(YES)
  )
  ARCHIVE_OUT(RUVARCH2
    LOCATION(REMOTE)
    COMMENT('REMOTE RPCV JOURNAL / RUV ARCHIVE')
    STATUS(INACTIVE)
    MUST_COMPLETE(NO)
  )
```

Using the RPCV_IN Keyword

Use the required RPCV_IN keyword on the ARCHIVE RPCV_IN command to identify the input RPCV journal. Set the value to the ddname of the journal in the JCL or to the fully qualified data set name.

Using the COMPLETE_COPY Keyword

Use the optional COMPLETE_COPY keyword on the ARCHIVE RPCV_IN command in the same way as you would use it on the ARCHIVE JOURNAL_IN command. For more information, see “Using the COMPLETE_COPY Keyword” on page 228.
Using Other Keywords

Use the required ARCHIVE_OUT keyword and its related optional keywords on the ARCHIVE RPCV_IN command in the same manner as you would use them on the ARCHIVE ARCHIVE_IN command. For more information, see “Using the ARCHIVE_OUT Keyword” on page 261.

Coding ARCHIVE RPCV_IN Statements

To code an ARCHIVE RPCV_IN statement, use the following syntax:

```
ARCHIVE RPCV_IN(ddn8_or_dsn44)
  { COMPLETE_COPY }
ARCHIVE_OUT(ddn8
  { LOCATION(locname8) }
  { COMMENT('comment fields') }
  { STATUS(ACTIVE | INACTIVE) }
  { MUST_COMPLETE(YES | NO) }
}
  { ARCHIVE_OUT(ddn8, ...) }
;
```
Printing Archive Data

At the request of Product Support, you can use the RUVZSM0 utility to print the data in an archive in hexadecimal format.

Using the PRINT ARCHIVE_IN Command

Use the PRINT ARCHIVE_IN command to print selected records from the archive, as shown in the following example:

```
PRINT ARCHIVE_IN(ARCHIVE1)
  COUNT(100, 500)
  SELECTION_EXIT(PROGRAM1)
;
```

**NOTE**

Use the REPORT(DETAIL) keyword on the SET or SET DEFAULT command to display the entire record. Use REPORT(SUMMARY) to display the first 10 bytes of each record.

Using the ARCHIVE_IN Keyword

Use the required ARCHIVE_IN keyword on the PRINT ARCHIVE_IN command to identify the archive data set. Set the value to the ddname in the JCL or to the fully qualified data set name.

Using the COUNT Keyword

Use the optional COUNT keyword on the PRINT ARCHIVE_IN command to select the range of records to print. Set the value to the relative record number of the first record to print (the *from* value) and to the relative record number of the last record to print (the *to* value).
Using the START_TIME keyword

The optional START_TIME keyword is valid with the PRINT ARCHIVE_IN command. You can use this keyword with the STOP_TIME keyword to select the range of records to print from the archive.

Use the START_TIME keyword to specify the start (earliest) timestamp of the range of selected records.

--- WARNING ---

Specify the COUNT keyword or the START_TIME keyword, but not both.

For more information, see “Using the START_TIME Keyword” on page 111.

Using the STOP_TIME keyword

The optional STOP_TIME keyword is valid with the PRINT ARCHIVE_IN command. You can use this keyword with the START_TIME keyword to select the range of records to print from the archive.

Use the STOP_TIME keyword to specify the stop (latest) timestamp of the range of selected records.

For more information, see “Using the RUV ISPF Interface” on page 113.

Using the SELECTION_EXIT keyword

You can use the SELECTION.EXIT keyword on the PRINT ARCHIVE_IN command to have RUV invoke a user exit routine for each record that it processes during the recovery. It can also be used to override a user exit selection name defined during an ADD or UPDATE VSAM_FILE. See “Using the SELECTION_EXIT User Exit” on page 517. As the value of the SELECTION_EXIT keyword, specify the program name of the exit routine.
Coding PRINT ARCHIVE_IN Statements

To code a PRINT ARCHIVE_IN statement, use the following syntax:

```
PRINT ARCHIVE_IN(dd8_or_dsn44)
  COUNT(from, to)
  SELECTION_EXIT(program8)
;
```

As shown in the following example, selected records are printed from the archive and the selection exit RUNTYPE receives the same data:

```
PRINT ARCHIVE_IN(ARCHIVE1)
  COUNT(100, 500)
  SELECTION_EXIT(RUNTYPE)
;
```
Working with Backups

This chapter describes how to create and register backups of files. The following information is included:

Backup Overview .............................................................. 279
Backup Methods .............................................................. 279
Backup Strategies .............................................................. 282
Backup Registration Records in the RUV Repository .................. 283
Considerations for Backups ............................................... 284
Integrating RUV in an Existing Backup Strategy .................... 285
Controlling Backup Environment Options ............................. 286
Creating RUV Backups ..................................................... 288
Overview of the RUV Backup Function ................................. 288
Using the RUV Backup Function ........................................ 289
Coding JCL for RUV Backups of Individual VSAM Files .......... 290
Coding JCL for RUV Backups of Individual Sequential Files ...... 294
Coding JCL for RUV Backups of VSAM Groups .................... 294
Coding JCL for RUV Backups of Sequential Groups .............. 297
RUV Backup Examples .................................................... 298
Creating BWO Backups .................................................... 300
Overview of BWO Backups .............................................. 300
Using the BWO Feature .................................................... 301
Creating Instant Snapshot Copies ...................................... 302
Overview of Instant Snapshot Copies .................................. 302
Creating an Instant Snapshot Copy ..................................... 303
Working with VSAM Name Models .................................... 304
Coding JCL to Create Instant Snapshot Copies ..................... 307
Instant Snapshot Copy Examples ........................................ 311
Using the RBA0 Backup Detection Feature ........................... 312
Overview of the RBA0 Backup Detection Feature .................. 312
Identifying an RBA0 Backup ............................................. 312
Implementing the RBA0 Backup Detection Feature ................. 313
Using SMS Backups and IDCAMS REPRO Copies .................. 314
Automatically Using SMS Backups .................................... 314
Automatically Using IDCAMS REPRO Copies ........................ 314
Creating External Backups ............................................... 315
Overview of External Backup Support ................................ 315
Implementing External Backup Support ........................................... 317
Example of External Backup Support ............................................ 318
Working with Job JCL Records for Backup and Restore Processes .......... 327
  Overview of Job JCL Records .................................................... 327
  Creating a Job JCL Record with the ISPF Interface ......................... 327
  Creating a Job JCL Record with the RUVZSM0 Utility ....................... 329
  Coding JCL to Work with Job JCL Records .................................. 329
Storing Information into a File ................................................... 332
  Coding the STORE INTO_FILE Command .................................... 332
  Example of STORE INTO_FILE Command Usage .............................. 333
Submitting Job JCL Records for Execution .................................... 334
  Overview of the Submit Function .............................................. 334
  Coding JCL to Submit Job JCL Records for Execution ..................... 341
Working with External Job Records .............................................. 343
  Coding REPORT and DELETE EXTERNAL_JOB Commands .................. 343
  EXTERNAL_JOB Report ......................................................... 344
Creating Copies of Backups ....................................................... 345
  Overview of the Copy Backup Function ..................................... 345
  Coding JCL for the COPY BACKUP Command ................................ 346
  Copy Backup Function Considerations ...................................... 351
  Copy Backup Function Example .............................................. 352
Explicitly Registering Backups ................................................... 354
  Overview of Backup Registration ............................................ 354
  Coding JCL for the REGISTER BACKUP_FILE Command .................... 355
  Coding JCL for the REGISTER RESTORE_JOB_JCL Command .............. 357
  Coding JCL for the REGISTER RESTORE_METHOD Command ............... 359
  Coding JCL for the REGISTER BACKUP_IN Command ....................... 362
Working with Backup Records in the Repository ............................. 363
  Overview of Update, Delete, and Report Backup Functions ................. 363
  Coding JCL for the UPDATE BACKUP_FILE Command ........................ 364
  Coding JCL for the DELETE BACKUP_FILE Command ........................ 367
  Coding JCL for the REPORT BACKUP_FILE Command ........................ 369
  Coding JCL for the REPORT BACKUP_METHOD Command .................... 371
Backup Overview

For recoverability of data, you must take periodic backups of files. A backup is simply a copy of the data. If the original file is lost or damaged, if the data is corrupted by a logic error, or if you need a duplicate copy of the file for testing, the file can be restored to its original condition by using the backup data set. (For complete information about how backups are used during restore processing, see Chapter 10, “Restoring Data Sets.”)

As explained in this chapter, RUV automates and optimizes the handling of backups. The following sections provide overview information about these topics:

- backup methods that RUV supports
- strategies that you can use to manage backups
- backup registration records in the repository
- considerations for backups

**NOTE**

For information about how to produce backups of the RUV repository, see Chapter 14, “Maintaining the RUV Repository.”

Backup Methods

RUV supports a wide variety of backup methods.

**NOTE**

When RUV automatically selects a backup to restore, it selects the backup that is the most current (the latest backup that has occurred before the target recovery time).

RUV Backups

You can use the RUV Backup function to create one or more copies of KSDS, ESDS, RRDS, VRRDS VSAM base clusters, and sequential files. RUV writes these RUV backups as sequential data sets on tape or DASD.

RUV automatically detects and restores RUV backups during restore and forward recovery processes. During a VSAM sphere backup, RUV can record enough information to allow it to automatically delete/define the file and rebuild any alternate indexes during a restore.

For more information, see “Creating RUV Backups” on page 288.
VSAM BWO Backups

If your CICS regions need constant access to your VSAM files, you can use the Backup-While-Open (BWO) feature of RUV to take BWO backups (also known as fuzzy backups). In contrast with a crisp backup, which is taken while the VSAM file is closed to all processes other than the backup process, a fuzzy backup is taken while the VSAM file remains open to transaction updates. A fuzzy backup is combined with relevant records of transaction updates during file recovery.

To create BWO backups, you implement the BWO feature in the environment, enable the BWO backup method, and execute the RUV Backup function for the file or group of files.

For more information, see “Creating BWO Backups” on page 300.

Instant Snapshot Copies

You can use the RUV Backup function to produce Instant Snapshot copies through the use of the BMC Software SNAPSHOT UPGRADE FEATURE (SUF) for a VSAM component, which works with the technology that is available with intelligent storage devices. You can also make an Instant Snapshot copy of a sequential file. An Instant Snapshot copy is a physical duplicate of a file without paths and alternate indexes. The copy process is nearly instantaneous, regardless of the size of the data set. An Instant Snapshot copy can also be restored in seconds, and RUV rebuilds paths. RUV automatically recognizes and handles an Instant Snapshot copy during restore and forward recovery processes.

For more information, see “Creating Instant Snapshot Copies” on page 302.

Sequential File Backups

RUV gives you the ability to backup and restore sequential files, as well as VSAM files.

With this feature, you can

- keep your sequential files generally in sync with your VSAM files
- have the backup information for all your important files stored in the RUV repository
RBA0 Backups

Through the RBA0 backup detection feature, RUV can detect that a VSAM file is empty and register a RBA0 backup at that point in time. It can also register an RBA0 backup when a file that has been defined with the REUSE option is opened for output. (This action causes the high used relative byte address to be reset to zero, which effectively deletes all records in the file.)

An RBA0 backup is simply a record in the repository; no physical data set is associated with an RBA0 backup. However, an RBA0 backup is intended to be used, in the same way as any other type of backup, as a starting point for recovery.

For more information, see “Using the RBA0 Backup Detection Feature” on page 312.

SMS and IDCAMS Backups

RUV can automatically handle existing Storage Management Subsystem (SMS) backups of files. You do not need to register SMS backups in the repository for RUV to select and restore them as appropriate for the situation. RUV does not invoke SMS to perform backups.

RUV can automatically restore backups that are produced with the IDCAMS utility REPRO command. You must use the REGISTER BACKUP_FILE command to register the backups in the RUV repository explicitly; you can set up the backup job to perform this registration automatically.

For more information, see “Using SMS Backups and IDCAMS REPRO Copies” on page 314.

External Backups

In addition to supporting backups that are produced by RUV, SMS, and IDCAMS, RUV fully supports and automates the use of external backups that are created by other processes and vendor products. RUV can submit the JCL that executes the external backup process and registers the external backup in the repository. During the restore process, RUV can submit the JCL that restores the file from the external backup or issue a request to the operator that provides the necessary information for the file to be restored manually.

For more information, see “Creating External Backups” on page 315.
Backup Strategies

The following topics relate to the strategies you can use for backups. For a complete discussion about general backup strategies that you might employ, see “Managing Backups” on page 70.

Group Backups and Individual File Backups

You can manage backups individually for each file, collectively for groups of VSAM files or groups of sequential files, or in any combination of individual and collective management that suits your environment. RUV groups simplify the setup of backup processes and ensure consistent activity for related files.

As much as possible, RUV performs backups for the files in a group in parallel, which minimizes the elapsed time for completing the backup process and maximizes the effective use of available resources. If you specify individual files for backup in a single job step, RUV performs the backups serially in the order that you specify them.

If you specify a group for backup, RUV resolves and expands the group into its component file names; this expansion is based on the file names that are recorded in the RUV repository at execution time. The expanded group of files can be backed up in parallel to DASD, multiple tape drives, or stacked tapes, providing you with superior tape usage.

Backup data sets that were created by a VSAM group backup process can be used for individual file restore or recovery, as well as for VSAM group restore and recovery. Backup data sets that were created by a sequential group backup process can be used for individual file restore, as well as for sequential group restore.

Most RUV backup management commands allow you to specify RUV groups. Information about using groups with those commands is included in this chapter. For more information about defining groups, see “Defining Groups” on page 601.

Copies of Backups

Depending on your backup strategy, you might need multiple copies of a backup data set (for example, you might want to keep one copy onsite for application recovery and send one copy offsite for disaster recovery), or you might need a copy in a different format (for example, you might want to create a traditional backup from an Instant Snapshot copy).

RUV can produce multiple copies of the backup data set during the backup process, register the copies, and mark the copies with a status that indicates their eligibility for restore processing.
You can also use the RUV COPY BACKUP command or the IDCAMS REPRO command to copy backup data sets in a separate process at any time after the backup process completes. For RUV to recognize and use copies that you create with the REPRO command, you must register them explicitly. RUV implicitly registers copies that you create with the COPY BACKUP command.

For more information, see “Creating Copies of Backups” on page 345.

**Backup Environment Options**

Backup environment options control how RUV handles backup-related concerns, such as which backup methods you want to enable, whether to perform incremental backups, and how long to keep backup registration records in the repository.

For more information, see “Controlling Backup Environment Options” on page 286

**Backup Registration Records in the RUV Repository**

To manage backups, RUV maintains backup registration records in the RUV repository.

**Registration of Backups**

For RUV to be able to use most types of backups (except SMS backups), the backup must be registered. RUV creates a backup registration record that contains all the information that RUV needs for identifying and using the backup, including the backup data set name, the data set name of the original file that was backed up, the time when the backup was created, and the backup method that was used to create the backup.

RUV registers RUV backups, Instant Snapshot copies, and RBA0 backups implicitly during the backup process. (Implicit registration means that you do not use the REGISTER command.) RUV also implicitly registers copies of backups that are created with the COPY BACKUP command. RUV detects and uses SMS backups automatically without a backup registration record in the repository.
For RUV to create a backup registration record in the repository for IDCAMS backups and external backups, an explicit REGISTER command is required. You can automate registration in the backup job by including a step to execute the REGISTER command. You can also use the REGISTER command as needed to create or recreate a backup registration record (for example, if a backup registration record has been deleted while the backup itself is still needed).

For more information, see “Explicitly Registering Backups” on page 354.

**Update, Delete, and Report Functions**

In most cases, RUV manages backup registration records in the repository automatically so that no manual tasks are necessary. In certain circumstances, however, you might want or need to work directly with backup registration records. You can use the following RUV commands for backup registration records:

- Use the UPDATE BACKUP_FILE command to change information in the backup registration record, including the status, location, comments, and registration time. For example, you might need to change a backup registration record from inactive to active status as you prepare for an offsite recovery.

- Use the DELETE BACKUP_FILE command to delete backup registration records. For example, you might need to delete a backup registration record manually if the backup that it describes has been lost or damaged. Normally, RUV deletes backup registration records when they are no longer needed based on repository purge criteria.

- Use the REPORT BACKUP_FILE command to obtain a report about the backups that are registered in the repository. The primary use of this report is to determine the results of a command before you execute it.

- Use the REPORT BACKUP_METHOD command to obtain a report about the backup methods that are enabled in your environment.

For more information, see “Working with Backup Records in the Repository” on page 363.

**Considerations for Backups**

RUV uses a relative byte address (RBA) to identify the records in a VSAM ESDS. Data compression products can change the RBA of an ESDS record as updates occur. Because of the unpredictability of compressed ESDS RBAs, you should exclude these data sets from compression if you want to recover them with RUV.
Integrating RUV in an Existing Backup Strategy

You probably can integrate RUV with your existing backup strategy with little or no effort. Table 13 shows how to do so. The first column lists a backup strategy that you might already employ or a need you might have. The second column outlines how you would use RUV with that strategy or to meet that need. The last column provides a reference to the section that provides more information.

Table 13  Integration of RUV into an Existing Backup Strategy

<table>
<thead>
<tr>
<th>Backup Strategy</th>
<th>Implementation of RUV</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>You already use SMS to perform backups of files.</td>
<td>Typically, no action is necessary for RUV to recognize and restore SMS backups.</td>
<td>314</td>
</tr>
<tr>
<td>You use IDCAMS to perform backups of VSAM files.</td>
<td>For RUV to recognize and restore an IDCAMS backup, you must use the REGISTER command to create a backup registration record.</td>
<td>314</td>
</tr>
<tr>
<td>You use in-house or third-party backup software to perform backups of files.</td>
<td>For RUV to recognize and restore an external backup, you must use the REGISTER command to create a backup registration record. You can also implement techniques for invoking the external backup from within RUV.</td>
<td>315</td>
</tr>
<tr>
<td>You want VSAM files to remain available for update by CICS transactions while the backup process is being performed.</td>
<td>Use the RUV Backup-While-Open (BWO) feature to create BWO backups.</td>
<td>300</td>
</tr>
<tr>
<td>You need to keep two or more files synchronized.</td>
<td>Use RUV group backups.</td>
<td>288</td>
</tr>
<tr>
<td>You need extremely rapid backups of files.</td>
<td>Use RUV to create Instant Snapshot copies.</td>
<td>300</td>
</tr>
<tr>
<td>You need automated recovery.</td>
<td>Use RUV backup techniques or register your existing backups.</td>
<td>300</td>
</tr>
</tbody>
</table>
Controlling Backup Environment Options

In addition to general options that can apply to any RUV job step, you can use the SET command and the SET DEFAULT command to control backup environment options. These options tell RUV how to handle backup-related concerns, such as which backup methods you want to enable and how long to keep backup registration records in the repository.

Backup Environment Options

You can use the following keywords on SET and SET DEFAULT commands to control backup environment options:

**BACKUP_METHOD**
Use the BACKUP_METHOD keyword to enable or disable the use of the various backup methods that RUV supports, including Backup-While-Open (BWO) backups, Instant Snapshot copies, RBA0 backups, and external backups. RUV backups are always enabled. The creation of SMS backups and external backups is not affected by this keyword; however, this keyword does control RUV use of these types of backups during restore and recovery processing. For more information, see “Using the BACKUP_METHOD Keyword” on page 126.

**BACKUP_CHANGED_ONLY and INCREMENTAL_BACKUP_SUPPORT**
RUV works with the VTOC flag that indicates whether the file has been changed since the last backup.

You can use the INCREMENTAL_BACKUP_SUPPORT keyword to control whether RUV performs a requested backup for an individual file if the data set has not been changed since the last backup and also to control whether RUV resets the flag at the end of the backup process to indicate that a backup has been performed. For more information, see “Using the INCREMENTAL_BACKUP_SUPPORT Keyword” on page 132.

If you use groups, you can use the BACKUP_CHANGED_ONLY keyword on the SET or SET DEFAULT command to control whether RUV performs requested backups of all files in the group or only the files that have changed since the last backup. For more information, see “Using the BACKUP_CHANGED_ONLY Keyword” on page 126.

**READER_TASKS**
Use the READER_TASKS keyword on the SET DEFAULT or SET command to control the maximum number of reader tasks (TCBs) that RUV can attempt to attach for reading input files. This option is used in situations where RUV can process input files simultaneously and you want to limit the number of input tape drives. The resources that you make available with this keyword can also prevent multitasking of the reader process. For more information, see “Using the READER_TASKS Keyword” on page 134.
DAYS or CYCLES

Use the DAYS or CYCLES keyword to control the retention period for backup information in the repository. The repository purge process removes backup registration records automatically based on the DAYS or CYCLES value. These keywords have no effect on the backup process, but they do control how long backup registration records are kept.

You can specify DAYS or CYCLES on ADD VSAM_FILE, ADD SEQ_FILE, UPDATE SEQ_FILE, and UPDATE VSAM_FILE commands as well as on SET and SET DEFAULT commands. For more information, see “Purging Repository Records” on page 579, “Using the CYCLES Keyword” on page 129 and “Using the DAYS Keyword” on page 131.

Default and Override Values

As you are implementing RUV in your backup environment, evaluate the default values of the backup environment options to ensure that they are suitable for the methods and strategies that you have chosen and for the majority of the backup job steps in your environment. To determine which default values are in effect in your environment, use the REPORT DEFAULT command, as described in Chapter 8, “Creating Reports.”

If one or more default values are not appropriate, use the SET DEFAULT command to change them as described in “Setting Default and Override Environment Options” on page 124.

**WARNING**

If you change a default option value with the SET DEFAULT command, you change it for all RUV job steps in the environment. Be careful not to change a value that another job step might need for correct processing.

As you are setting up a new backup job step, evaluate whether the default values for backup environment options are appropriate for the job step. If they are not, you can set override values for the options by coding the SET command in the SYSIN control statement data set in the backup job step as described in “Setting Default and Override Environment Options” on page 124.
Creating RUV Backups

The following sections describe how to create RUV backups with the RUV Backup function.

Overview of the RUV Backup Function

To create RUV backups, you code and run the JCL to execute the RUV Backup function of the RUVZSM0 utility. In the control statement data set, you can specify individual files for backup with the BACKUP VSAM_IN or BACKUP_SEQ_IN command. You can specify a predefined group of files for backup with the BACKUP VSAM_GROUP or BACKUP SEQ_GROUP command.

Input and Output

As input, the Backup function supports sequential files, KSDS, ESDS, RRDS, and VRRDS VSAM base clusters. As output, the function writes one or more backup copies as sequential data sets. (In contrast, an Instant Snapshot copy creates a VSAM file without any alternate indexes.)

For VSAM files, if you specify the PREFIX(YES) keyword on the BACKUP statement, RUV writes each backup data set with a prefix. This prefix records physical information about the original VSAM base cluster. PREFIX(YES) is required for ESDS, RRDS, and VRRDS clusters. If you specify PREFIX(NO), which is valid for KSDSs only, RUV creates a flat-file backup that is identical to a copy that is created with the IDCAMS utility REPRO command.

For sequential files, all backups are not prefixed, no matter how the keyword is coded.

RUV can write backups to tape or DASD. RUV can stack backups for multiple files on a single tape volume (if space permits); stacking reduces the number of tapes that you must manage and store.

Registration

RUV implicitly registers an RUV backup by creating a backup registration record in the repository; it is not necessary to specify the REGISTER BACKUP command (as is required for registration of some other types of backups).

RUV automatically detects and restores RUV backups during restore and forward recovery processes. From the backup registration record in the repository, RUV obtains the timestamp relationship that it needs to select the correct archive files during recovery.
Using the RUV Backup Function

To use the RUV Backup function to create RUV backups, perform the following steps:

1. Determine which files to include for backup.

   The Backup function can select individual files, or it can process all of the files in a group that you have defined with the ADD or UPDATE VSAM_GROUP_DEFINITION or SEQ_GROUP_DEFINITION command (as described in “Defining Groups” on page 601).

2. Code the JCL to execute the Backup function of the RUVZSM0 utility.

   The JCL statements for executing the Backup function of the RUVZSM0 utility differ depending on whether you request a backup for individual files or for a defined group. To simplify explanations, this manual describes the JCL for individual file backups separately from the JCL for group backups; however, you can code a mixture of backup requests for individual file and groups in the same job step.

   - You request backups for individual files with the BACKUP VSAM_IN or BACKUP SEQ_IN command, as described in “Coding JCL for RUV Backups of Individual VSAM Files” on page 290 and “Coding JCL for RUV Backups of Individual Sequential Files” on page 294.

   - You request backups for a defined group with the BACKUP VSAM_GROUP or BACKUP SEQ_GROUP command, as described in “Coding JCL for RUV Backups of VSAM Groups” on page 294 and “Coding JCL for RUV Backups of Sequential Groups” on page 297.

3. Submit the JCL for execution.

4. Check the results of the execution.
Figure 50 shows model JCL for executing the Backup function for individual files, as specified with the BACKUP VSAM_IN command. See member BACKUP of the sample library for a sample job that you can copy and modify.

**Figure 50  BACKUP VSAM_IN Model JCL**

```
//jobname JOB job statement
//BACKUP EXEC PGM=RUVZSM0
//STEPLIB DD DISP=SHR,DSN=yourname.LOAD
//SYSPRINT DD SYSOUT=*  
//* REQUIRED DD STATEMENT FOR OUTPUT BACKUP DATA SET
//bkupout1 DD DISP=(NEW,CATLG),DSN=bkupout1.dsn ...
//*  
//* OPTIONAL DD STATEMENT FOR ADDITIONAL COPY OF BACKUP DATA SET
//bkupout2 DD DISP=(NEW,CATLG),DSN=bkupout2.dsn ...
//*  
//* IF DDNAME IS CODED FOR VSAM_IN, REQUIRED DD STATEMENT FOR
//* INPUT VSAM FILE
//vsamin DD DISP=SHR,DSN=vsamin.dsn
//*  
//SYSIN DD *
/* OPTIONAL SET COMMAND TO CONTROL ENVIRONMENT OPTIONS
SET keywords;  
/* REQUIRED BACKUP COMMAND TO INVOKE BACKUP FUNCTION FOR
* INDIVIDUAL VSAM FILE
BACKUP VSAM_IN(ddn_or_dsn)
  { PREFIX(YES | NO)
  { INVENTORY_ONLY(NO | YES)
  { COMMENT (COMMENT_FIELDS) }
  BACKUP_OUT(dsn8)
  { LOCATION(locname8) }
  { COMMENT(comment_fields) }
  { STATUS(ACTIVE | INACTIVE) }
  }
  BACKUP_OUT(ddn8 ...) }
;  
/*
```

The following information applies specifically to RUVZSM0 job steps that contain the BACKUP VSAM_IN command. For information about the standard DD statements and control statements that are common to all RUVZSM0 jobs, see “Using the RUV Batch Interface” on page 79.
Input VSAM Files

For backup of individual VSAM files, RUV can dynamically allocate the input VSAM files to be copied. You do not need to provide DD statements for these files if you specify the data set name as the value of the VSAM_IN keyword on the BACKUP VSAM_IN command. If you provide a DD statement for the input VSAM file, specify the ddname of the DD statement as the value of the VSAM_IN keyword.

Output Backup Data Sets

If you are requesting backups for individual files, you must specify a DD statement to describe each output backup data set. The ddname must match the value of the BACKUP_OUT keyword on the BACKUP VSAM_IN command.

If the backup data sets will be written to tape, consider stacking multiple backup data sets on the tape. If you use stacking, you might want to perform backups for VSAM groups instead of for individual VSAM files. The STACKED_TAPE_COUNT keyword on the ADD or UPDATE OUTPUT_MODEL command automates the tasks of specifying and maintaining tape stacks for the group. Using this keyword is much faster and easier than specifying and maintaining tape stacks manually.

By default, RUV uses the DCB attributes shown in Table 14 to allocate the backup data set.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLKSIZE</td>
<td>The RUV default depends on the device:</td>
</tr>
<tr>
<td></td>
<td>■ for a 3390 device, RUV uses 27998 (allowing 2 physical blocks per track)</td>
</tr>
<tr>
<td></td>
<td>■ for a tape device, RUV uses 32760</td>
</tr>
<tr>
<td></td>
<td>For blocksizes greater than these defaults, RUV writes one block per track.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can override the RUV default blocksize, but that action is not recommended.</td>
</tr>
<tr>
<td>LRECL</td>
<td>By default, RUV uses the LRECL attribute as documented in the IBM JCL reference documentation. You can override the LRECL to a size that is compatible with the record size of the input VSAM data set.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Overriding the LRECL is not recommended for prefixed backups.</td>
</tr>
<tr>
<td>RECFM</td>
<td>RUV uses VBS as the default value. You can override it to any format (V, VS, VB, VBS, F, FB) that is compatible with the record format of the input VSAM data set.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Overriding the RECFM is not recommended for prefixed backups. All prefixed backups must be variable, not fixed.</td>
</tr>
<tr>
<td>DSORG</td>
<td>RUV uses PS as the default value. Do not change this value.</td>
</tr>
</tbody>
</table>
**BACKUP VSAM_IN Command**

To create one or more RUV backups for an individual VSAM file, you code one BACKUP VSAM_IN command and its related keywords in the SYSIN data set. You can include multiple BACKUP VSAM_IN commands to create backups for multiple individual VSAM files.

Keywords on the BACKUP VSAM_IN command identify the file to back up and the options to use for the backup process. The following sections describe the valid keywords in the order that they are listed in the model JCL in Figure 50.

You can also control options in the backup environment with keywords on the SET or SET DEFAULT command. For more information, see “Controlling Backup Environment Options” on page 286.

For information about the syntax of RUVZSM0 control statements, see “Command Language Elements” on page 82.

**VSAM_IN Keyword**

Use the required VSAM_IN keyword on the BACKUP VSAM_IN command to identify the individual file to back up. You can code only one VSAM_IN keyword on each BACKUP VSAM_IN command. Set the value to the ddname of the DD statement that describes the input VSAM data set in the JCL, or set it to the fully qualified data set name of the VSAM file to have RUV dynamically allocate the data set. Wild cards are not allowed.

**PREFIX Keyword**

You can use the optional PREFIX keyword on the BACKUP VSAM_IN command to control whether RUV creates a prefixed backup of a KSDS.

Prefixed backup files contain a prefix on each record. The restore process and the forward recovery process use the prefix information to manage the synchronization of all activity against the master data set.

Prefixed backup files contain the VSAM sphere definition along with the VSAM data. The prefix allows RUV to rebuild the VSAM sphere as part of the restore or recovery process as needed. A separate IDCAMS delete/define step is not needed unless you want to change the sphere definition during the restore. RUV performs recovery through the base cluster only, and not through paths.

RUV always creates prefixed backups for ESDSs and RRDSs. For KSDSs, you can use the PREFIX keyword to control whether RUV creates a prefixed backup. It is recommended that you allow RUV to add the prefix.
The record count and byte count values reported in the output from the backup job reflect the values pertaining to the input VSAM file. They do not account for the additional prefixed bytes and records written to the backup copy.

Set one of the following values for the PREFIX keyword. The default value is YES.

YES  Add the prefix to each record in the backup file.

NO  Do not add the prefix. This value is valid for a KSDS backup only. If you specify this value for a file other than a KSDS, the task fails.

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the BACKUP VSAM_IN command to produce a report of the resources without actual execution of the backup process. Because no execution occurs, no data sets are cataloged if you use this option.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

NO  Produce a backup without a preliminary resource report.

YES  Produce a preliminary report of VSAM and backup file resources, but do not perform the backup process.

BACKUP_OUT Keyword

Use the required BACKUP_OUT keyword on the BACKUP VSAM_IN command to identify the data set to contain the backup of the VSAM data set. The backup contains a copy of all records from the input VSAM file, along with the control information about the file. At completion of the backup process, RUV registers the information about the backup file in the repository.

Set the value of the BACKUP_OUT keyword to the ddname that identifies the output data set in the JCL. For more information about coding the DD statement, see “Output Backup Data Sets” on page 291.

Code these keywords and their values after the ddname value but before the closing parenthesis for the BACKUP_OUT value. You can code multiple BACKUP_OUT keywords to request multiple copies of the output backup data set.
Other Optional Keywords

You can use the following optional keywords with the BACKUP VSAM_IN command, the BACKUP_OUT keyword, or the OUTPUT_MODEL keyword to set options for handling the backup data set:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)

NOTE

If you use the COMMENT keyword on the general BACKUP VSAM_IN statement, it is propagated to all COMMENT fields that are generated unless you code the COMMENT keyword on the specific BACKUP_OUT keyword.

Coding JCL for RUV Backups of Individual Sequential Files

This process works exactly the same as the process for “Coding JCL for RUV Backups of Individual VSAM Files” on page 290 with the following exceptions:

- The keyword SEQ_IN is used in place of VSAM_IN on the BACKUP command. This keyword is where you name the sequential file that is being backed up. This keyword treats sequential files exactly the same as the VSAM_IN keyword treats VSAM files.

- The keyword PREFIX is ignored. All sequential file backups are not prefixed, no matter how the keyword is coded.

Coding JCL for RUV Backups of VSAM Groups

Figure 51 on page 295 shows model JCL for executing the Backup function for VSAM groups, as specified with the BACKUP VSAM_GROUP command.

NOTE

Before the BACKUP VSAM_GROUP command is executed, the VSAM group must be defined with the ADD VSAM_GROUP_DEFINITION or UPDATE VSAM_GROUP_DEFINITION command.
The following information applies specifically to RUVZSM0 job steps that contain the BACKUP VSAM_GROUP command. For information about the standard DD statements and control statements that are common to all RUVZSM0 jobs, see “Using the RUV Batch Interface” on page 79.

### Input VSAM Files

For a VSAM group, RUV always dynamically allocates the input VSAM files to be copied. The data set names are identified in the VSAM group definition. In the backup JCL, you do not provide DD statements for these files.

### Output Backup Data Sets

For VSAM groups, RUV dynamically allocates the output data set for each requested backup copy from the information that you specify with the OUTPUT_MODEL keyword on the BACKUP VSAM_GROUP command. For more information, see “OUTPUT_MODEL Keyword” on page 297.

### BACKUP VSAM_GROUP Command

To create one or more RUV backups for a defined group of VSAM files, you code one BACKUP VSAM_GROUP command and its related keywords in the SYSIN data set. You can include multiple BACKUP VSAM_GROUP commands to create backups for multiple VSAM groups.
Keywords on the BACKUP VSAM_GROUP command identify the group to back up and the options to use for the backup process. The following sections describe the valid keywords in the order that they are listed in the model JCL in Figure 50.

You can also control options in the backup environment with keywords on the SET or SET DEFAULT command. For more information, see “Controlling Backup Environment Options” on page 286.

For information about the syntax of RUVZSM0 control statements, see “Command Language Elements” on page 82.

**VSAM_GROUP Keyword**

Use the required VSAM_GROUP keyword on the BACKUP VSAM_GROUP command to identify the VSAM group to back up. You can code only one VSAM_GROUP keyword on each BACKUP VSAM_GROUP command. Set the value to the name of the previously defined VSAM group.

For more information about defining VSAM groups, see “Defining Groups” on page 601.

**PREFIX Keyword**

You can use the optional PREFIX keyword on the BACKUP VSAM_GROUP command to control whether RUV creates a prefixed backup of a KSDS.

For more information, see “PREFIX Keyword” on page 292.

**INVENTORY_ONLY Keyword**

You can use the optional INVENTORY_ONLY keyword on the BACKUP VSAM_GROUP command to produce a report of the resources without actual execution of the backup process. Because no execution occurs, no data sets are cataloged if you use this option.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

|- NO | Produce a backup without a preliminary resource report. |
|- YES | Produce a preliminary report of VSAM and backup file resources, but do not perform the backup process. |
OUTPUT_MODEL Keyword

Use the OUTPUT_MODEL keyword on the BACKUP VSAM_GROUP command to identify the output model that contains the allocation information that you want to use for the output backup data set. The keyword value is the name of a previously defined output model.

If you want to create multiple copies of the backup data set, code the OUTPUT_MODEL keyword once for each copy.

You can define an output model in the same job step that executes the Backup function, but typically you would define output models at the same time that you define VSAM groups. For more information, see “Using the ADD OUTPUT_MODEL Command” on page 615.

Other Optional Keywords

You can use the following optional keywords with the OUTPUT_MODEL keyword to set options for handling the backup data set:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)

Coding JCL for RUV Backups of Sequential Groups

This process works exactly the same as the process for “Coding JCL for RUV Backups of VSAM Groups” on page 294 with the following exceptions:

- The keyword SEQ_GROUP is used in place of VSAM_GROUP on the BACKUP command. This keyword is where you name the sequential files that are being backed up. This keyword treats sequential files exactly the same as the VSAM_GROUP keyword treats VSAM files.

- The keyword PREFIX is ignored. All sequential file backups are not prefixed, no matter how the keyword is coded.
RUV Backup Examples

The following example shows the creation of backups for several individual VSAM files and for a defined group of VSAM files. Assume that the VSAM group and the output models previously have been defined.

Figure 52  BACKUP Examples

```
//RUVBKUPJ JOB job statement
//*
//RUVBKUP EXEC PGM=RUVZSM0
//STEPLIB DD DISP=SHR,DSN=yourname.LOAD
//VSAM01 DD DISP=SHR,DSN=MY.VSAM01.FILE
//BKUP01A DD DISP=(NEW,CATLG),DSN=MY.BKUP01A.DATASET,
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE)
//BKUP01B DD DISP=(NEW,CATLG),DSN=MY.BKUP01B.DATASET,
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE)
//BKUP02 DD DISP=(NEW,CATLG),DSN=MY.BKUP02.DATASET,
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE)
//SYSPRINT DD SYSOUT=*       
//SYSIN DD *                  
SET SUBSYSTEM(TEST);
*----------------------------------------------------*
BACKUP VSAM_IN(VSAM01)
   COMMENT("BACKUP TEST")
   BACKUP_OUT(BKUP01A LOCATION("VAULT") STATUS(ACTIVE))
   BACKUP_OUT(BKUP01B LOCATION("OFFSITE") STATUS(INACTIVE));
*----------------------------------------------------*
BACKUP VSAM_IN(MY.VSAM02.FILE)
   BACKUP_OUT(BKUP02);
   PREFIX(NO)
*----------------------------------------------------*
BACKUP VSAM_GROUP(MYGROUP)
   OUTPUT_MODEL(TAPE LOCATION("VAULT") STATUS(ACTIVE))
   OUTPUT_MODEL(TAPE LOCATION("OFFSITE") STATUS(INACTIVE));
/*
```

- The SET command tells RUV to use the TEST RUV subsystem.
- The first BACKUP command tells RUV to create two backup copies of the VSAM file that is identified with the VSAM01 DD statement. Both copies are written with a prefix because the PREFIX keyword is not specified and the default value is YES. The first copy, which is written to the data set that is identified with the BKUP01A DD statement, is for the tape vault and has an active status. The second copy, which is written to the data set that is identified with the BKUP01B DD statement, is for offsite disaster recovery and has an inactive status. The comment “BACKUP TEST” is propagated to all COMMENT fields for this VSAM file.
The second BACKUP command tells RUV to create one backup copy of the VSAM file that has the data set name MY.VSAM02.KSDS. RUV dynamically allocates the input VSAM file, and it writes the backup to the data set that is identified with the BKUP02 DD statement. The copy is written with no prefix (it is assumed that the input file is a KSDS). The backup has an active status because the STATUS field is not specified.

The third BACKUP command tells RUV to create two backup copies of each VSAM file that is included in the VSAM group named MYGROUP. RUV dynamically allocates the input files from information in the group definition, which identifies the data set names of the files in the group. All copies are written with a prefix because the PREFIX keyword is not specified and the default value is YES. The first copy, which is written with the allocation parameters that are specified in an output model named TAPE, is for the tape vault and has an active status. The second copy, which is written with the same allocation parameters as the first copy, is for offsite disaster recovery and has an inactive status.
Creating BWO Backups

When you use the Backup-While-Open (BWO) feature, RUV can create a backup of a VSAM file while the file remains open to CICS. The following sections describe how to create BWO backups.

Overview of BWO Backups

A BWO backup is also known as a fuzzy backup. In contrast with a crisp backup, which is taken while the VSAM file is closed to all processes other than the backup process, a fuzzy backup is taken while the VSAM file remains open to transaction updates. A fuzzy backup is combined with relevant records of transaction updates during file recovery. For more information about crisp and fuzzy backups, see “Types of Backups Needed for a Recovery” on page 44.

In format and structure, a BWO backup is identical to an RUV backup. You use the same JCL to create a BWO backup as you would use to create an RUV backup. The difference is in the backup registration record; if the BWO feature is active and enabled, and if the VSAM file is open to CICS, RUV registers the backup as a BWO backup rather than an RUV backup. This registration information notifies RUV that transaction updates might need to be reapplied at restore or recovery time.
Using the BWO Feature

To create BWO backups, perform the following steps:

1. Implement the BWO feature in the environment.

Typically, the person who installs RUV implements the BWO feature by performing the following types of tasks during installation:

- defining RUV transactions and programs to CICS
- customizing CICS startup and shutdown by adding the appropriate entries to the PLTP1 table and the PLTSD table and adding RUVPLTxx members into DFHPLT assemblies
- verifying that required parameters are specified in the system initialization table (SIT)
- updating CICS startup JCL to include the RUV load library or libraries
- coding the CICS file definitions
- authorize the RUV load libraries

For more information about how to implement and control the BWO feature, see the Recovery Utility for VSAM Installation and Customization Guide.

1. Code the BWO value for the BACKUP_METHOD keyword on the SET or SET DEFAULT command.

This keyword enables the creation of BWO backups. If you set this keyword on the SET DEFAULT command, you do not need to set it for every backup or copy job step. For more information, see “Using the BACKUP_METHOD Keyword” on page 126.

2. Allow CICS to open the file or group of files to be copied.

3. Code JCL to execute the RUV Backup function for the file or group of files.

This JCL is identical to the JCL for executing the RUV Backup function to create RUV backups. For details, see “Coding JCL for RUV Backups of Individual VSAM Files” on page 290 and “Coding JCL for RUV Backups of VSAM Groups” on page 294.

4. Execute the JCL, and check the job completion code. If the job completes with any code other than 0, review the messages that were issued during the job.
Creating Instant Snapshot Copies

The following sections describe how to create and use Instant Snapshot copies.

Overview of Instant Snapshot Copies

To produce Instant Snapshot copies, RUV uses the BMC Software SNAPSHOT UPGRADE FEATURE (SUF) for VSAM component to create a duplicate of a VSAM or sequential data set in seconds by exploiting the technology that is available with intelligent storage devices. The copy process is nearly instantaneous, regardless of the size of the data set. An Instant Snapshot copy can also be restored in seconds. RUV automatically recognizes and handles an Instant Snapshot copy during the restore process.

SUF, XBM, and the XBM Subsystem

To create Instant Snapshot copies, you must install, customize, and implement SUF. This component is packaged for distribution with the BMC Software EXTENDED BUFFER MANAGER (XBM) product. SUF for VSAM is included with RUV at no additional charge. SUF requires an active XBM subsystem for creating Instant Snapshot copies and for performing an instant restore from an Instant Snapshot copy.

Instant Snapshot Copies

An Instant Snapshot copy is a duplicate of the original data set without paths (if any). For example, if the input data set is a KSDS, the Instant Snapshot copy is also a KSDS. Instant Snapshot copies must be made on the same type of intelligent storage device as the device containing the original data sets to be copied. The copy and original data set can reside on different physical devices.

VSAM Name Models

RUV must construct the data set name of an Instant Snapshot copy. You create a VSAM name model (as a record in the RUV repository) to contain the required information for constructing the name of a VSAM or sequential file. SUF automatically manages all other characteristics of the Instant Snapshot copy.
SUF Documentation

For information about the documents and media you need for understanding, installing, customizing, and implementing SUF and managing the XBM subsystem, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE Release Notes.

Error Messages

RUV and SUF communicate with each other about any detected errors. If SUF detects the error, it sets a reason code that RUV reports in an RUV message. For an explanation of the reason code, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide. You should also examine the XBM SYSPRINT or JES log for SUF messages that indicate the cause of and solution for the problem.

Creating an Instant Snapshot Copy

After you have installed and implemented SUF, perform the following steps to create an Instant Snapshot copy:

1. Code the SNAPSHOT value for the BACKUP_METHOD keyword on the SET or SET DEFAULT command.

   This keyword enables the creation of Instant Snapshot copies. If you set this keyword on the SET DEFAULT command, you do not need to set it for every backup or copy job step. For more information, see “Using the BACKUP_METHOD Keyword” on page 126.

2. Code the XBMID keyword on the SET or SET DEFAULT command.

   This keyword identifies the XBM subsystem to use for SUF. If you set this keyword on the SET DEFAULT command, you do not need to set it for every backup or copy job step. For more information, see “Using the XBMID Keyword” on page 141.

3. Create one or more VSAM name models with the ADD VSAM_NAME_MODEL command.

   This command defines the VSAM name model for RUV to use when it creates an Instant Snapshot copy of either a VSAM file or a sequential file. If your organization uses standard naming conventions, you can create VSAM name models with masks (using wildcard characters * and ?) that allow general application to most, if not all, of the files that you want to manage. If you predefine the models, you do not need this command for every backup or copy job step. For more information, see “Working with VSAM Name Models” on page 304.
4 Code the JCL to execute the Backup function or the Copy function of the RUVZSM0 utility:

- Use the BACKUP command to create the copy and register it as a backup in the RUV repository.
- Use the COPY command to create the copy but do not register it as a backup in the RUV repository.

On either command, the VSAM_NAME_MODEL keyword indicates that you want to create an Instant Snapshot copy and identifies the VSAM name model that you want to use to construct the data set name. For more information, see “Coding JCL to Create Instant Snapshot Copies” on page 307.

5 Make the XBM load modules available to the utility job step that performs the Instant Snapshot copy. You can do so by adding the XBM load library to the STEPLIB or JOBLIB concatenation in the job step JCL or by including the modules in the system LNKLST. The load modules must be APF-authorized.

6 Make sure that the data set to be copied is not allocated explicitly by any job step in the job JCL or ANY OTHER JOB. It must be dynamically allocated in the Instant Snapshot copy step. The input and output data set names must be coded in the SYSIN DD statement.

7 Submit the JCL for execution.

8 Check the results of the execution.

---

**Working with VSAM Name Models**

You use a VSAM name model to tell RUV how to build the data set name for an Instant Snapshot copy. A VSAM name model is a defined record in the RUV repository. You can use the RUVZSM0 utility to work with VSAM name models as follows:

- Create a VSAM name model with the ADD VSAM_NAME_MODEL command.
- Modify an existing VSAM name model with the UPDATE VSAM_NAME_MODEL command.
- Delete one or more VSAM name models with the DELETE VSAM_NAME_MODEL command.
- Obtain a report about the information that is stored in one or more VSAM name models with the REPORT VSAM_NAME_MODEL command.
ADD VSAM_NAME_MODEL Command

Use the ADD VSAM_NAME_MODEL command to create a VSAM name model for RUV to use when it creates an Instant Snapshot copy. Use the following syntax:

```
ADD VSAM_NAME_MODEL(name32)
  VSAM_FILE_MASK(dsn_mask44, dsn.variables)
    | REPLACE(NO | YES))
    | COMMENT(comment_fields))
    | STATUS(ACTIVE | INACTIVE))
;  
```

UPDATE VSAM_NAME_MODEL Command

Use the UPDATE VSAM_NAME_MODEL command to change an existing VSAM name model. Use the following syntax:

```
UPDATE VSAM_NAME_MODEL(name32)
  VSAM_FILE_MASK(dsn_mask44, dsn.variables)
    | REPLACE(NO | YES))
    | COMMENT(comment_fields))
    | STATUS(ACTIVE | INACTIVE))
;  
```

DELETE VSAM_NAME_MODEL Command

Use the DELETE VSAM_NAME_MODEL command to delete one or more existing VSAM name models. Use the following syntax:

```
DELETE VSAM_NAME_MODEL(name32,*,....)
;  
```

REPORT VSAM_NAME_MODEL Command

Use the REPORT VSAM_NAME_MODEL command to obtain information about one or more existing VSAM name models. Use the following syntax:

```
REPORT VSAM_NAME_MODEL(name32,*,....)
;  
```

VSAM_NAME_MODEL Keyword

The VSAM_NAME_MODEL keyword defines a unique identifier for a VSAM name model. You can use the VSAM_NAME_MODEL keyword on the ADD, UPDATE, DELETE, and REPORT commands to identify the model that you want to work with.
On the ADD or UPDATE command, specify a single specific user-defined name (1–32 characters) as the value of the VSAM_NAME_MODEL keyword value. On the DELETE or REPORT command, you can specify multiple values, each of which can be a specific name or a mask (using the * and ? wildcard characters).

**VSAM_FILE_MASK Keyword**

The VSAM_FILE_MASK keyword defines the mask for RUV to use when it constructs the data set name of the Instant Snapshot copy for both a VSAM file or a sequential file.

You can code multiple instances of the VSAM_FILE_MASK keyword on the ADD or UPDATE VSAM_NAME_MODEL command. Each instance identifies one or more input VSAM files or sequential files for which the mask should apply when RUV constructs output copies of those files. The order in which you code the VSAM_FILE_MASK keywords is important because RUV uses the first mask that matches the input data set name.

The VSAM_FILE_MASK keyword has two values:

- The first value (dsn_mask44) identifies the input VSAM or sequential file or files for which the mask should apply. Specify a specific name (using all literal values) or a generic name (using any combination of literal values and wildcard characters * and ?). If the name of the data set to be copied matches the first value, RUV selects this mask for constructing the data set name of the output copy.

- The second value (dsn.variables) specifies the output data set name mask. You can specify any combination of literal values, substitution variables (user-defined, RUV-defined, and system-defined variables), and node separators that results in a valid data set name. For more information, see “Creating User Variables” on page 171 and “Using Variables” on page 173.

When you are using multiple VSAM_FILE_MASK keywords, be sure to code them in the order of the most specific to the least specific. RUV processes them in the order that you code them.

**REPLACE Keyword**

Use the REPLACE keyword on the ADD or UPDATE VSAM_NAME_MODEL command to specify whether RUV should replace an existing data set with the new copy. Specify one of the following values. The default value is NO.

- **NO**  If a data set with the constructed data set name already exists in the catalog, do not replace it with the new copy.

- **YES**  If a data set with the constructed data set name already exists in the catalog, replace it with the new copy.
VSAM Name Model Examples

The following example of a VSAM name model specifies two VSAM file masks. The first mask applies to all input VSAM data sets that have the high-level qualifiers PAYROLL.TAXWTHLD. The second mask applies to all input VSAM data sets that have the high-level qualifier PAYROLL.

```
ADD VSAM_NAME_MODEL(PAYROLLBACKUP1)
  VSAM_FILE_MASK(PAYROLL.TAXWTHLD.*,
                  TAXWTHLD.<NODE3>.,<RUVDAT>.,BACKUP1)
  VSAM_FILE_MASK(PAYROLL.*,,
                  <NODE1>.,<NODE2>.,<RUVDAT>.,BACKUP1)
```

This VSAM name model is used in an RUV job that produces copies of the following VSAM data sets:

```
PAYROLL.W200838.WKHOURS
PAYROLL.W200839.WKHOURS
PAYROLL.EMPLSTAT.TXYR2008
PAYROLL.TAXWTHLD.TXYR2008
PAYROLL.TAXWTHLD.TXYR2009
```

RUV constructs output data set names for each of these data sets as follows:

```
PAYROLL.W200838.D2008289.BACKUP1
PAYROLL.W200839.D2008289.BACKUP1
PAYROLL.EMPLSTAT.D2008289.BACKUP1
TAXWTHLD.TXYR2008.D2008289.BACKUP1
TAXWTHLD.TXYR2009.D2008289.BACKUP1
```

Coding JCL to Create Instant Snapshot Copies

Figure 50 shows model JCL for creating Instant Snapshot copies with several variations of the BACKUP command and the COPY command.

**NOTE**

Each BACKUP or COPY command can create one Instant Snapshot copy for each input VSAM file. If you want to create multiple Instant Snapshot copies of an input file, code multiple BACKUP or COPY commands.
Figure 53  Instant Snapshot Copy JCL

```jcl
//jobname JOB job statement
//BACKUP EXEC PGM=RUZZSM0
//STEPLIB DD DISP=SHR,DSN=yourname.LOAD
//SYSPRINT DD SYSOUT=*  
/*
** IF DDNAME IS CODED FOR VSAM_IN, REQUIRED DD STATEMENT FOR
** INPUT VSAM FILE
//vsamin DD DISP=SHR,DSN=vsamin.dsn
/*
//SYSIN DD *
* OPTIONAL SET COMMAND TO CONTROL ENVIRONMENT OPTIONS
SET BACKUP_METHOD(SNAPSHOT) XBMID(XBMT) other keywords;
*
* BACKUP COMMAND TO INVOKE BACKUP FUNCTION FOR SEQ FILE
BACKUP SEQ_IN(ddn_or_dsn)
  { INVENTORY_ONLY (NO | YES) }
  { COMMENT (COMMENT_FIELDS) }
  VSAM_NAME_MODEL(name32)
;
* BACKUP COMMAND TO INVOKE BACKUP FUNCTION FOR SEQ GROUP
BACKUP SEQ_GROUP(ddn_or_dsn)
  { INVENTORY_ONLY (NO | YES) }
  { COMMENT (COMMENT_FIELDS) }
  VSAM_NAME_MODEL(name32)
;
* BACKUP COMMAND TO INVOKE BACKUP FUNCTION FOR INDIVIDUAL VSAM FILE
BACKUP VSAM_IN(ddn_or_dsn)
  { PREFIX(YES | NO) }
  { INVENTORY_ONLY (NO | YES) }
  { COMMENT (COMMENT_FIELDS) }
  VSAM_NAME_MODEL(name32)
;
* COPY COMMAND TO INVOKE COPY FUNCTION FOR INDIVIDUAL VSAM FILE
COPY VSAM_IN(ddn_or_dsn)
  { PREFIX(YES | NO) }
  { INVENTORY_ONLY (NO | YES) }
  { COMMENT (COMMENT_FIELDS) }
  VSAM_NAME_MODEL(name32)
;
* COPY COMMAND TO INVOKE BACKUP FUNCTION FOR VSAM GROUP
COPY VSAM_GROUP(ddn_or_dsn)
  { PREFIX(YES | NO) }
  { INVENTORY_ONLY (NO | YES) }
  { COMMENT (COMMENT_FIELDS) }
  VSAM_NAME_MODEL(name32)
;
//*
```
BACKUP SEQ_IN Command

Use the BACKUP SEQ_IN command to create an Instant Snapshot copy of an individual sequential file and register the copy in the repository. The keywords that you can code on this command are almost identical to the keywords that you would code to create an RUV backup of an individual file, as explained in “Coding JCL for RUV Backups of Individual Sequential Files” on page 294. However, do not code the BACKUP_OUT keyword; instead, code the VSAM_NAME_MODEL keyword to identify the VSAM name model that tells RUV how to allocate the Instant Snapshot copy, as explained in “VSAM_NAME_MODEL Keyword” on page 310.

BACKUP SEQ_GROUP Command

Use the BACKUP SEQ_GROUP command to create an Instant Snapshot copy of a defined group of sequential files and register the copy in the repository. The keywords that you can code on this command are almost identical to the keywords that you would code to create an RUV backup of a sequential group, as explained in “Coding JCL for RUV Backups of Sequential Groups” on page 297. However, do not code the OUTPUT_MODEL keyword; instead, code the VSAM_NAME_MODEL keyword to identify the VSAM name model that tells RUV how to allocate the Instant Snapshot copy, as explained in “VSAM_NAME_MODEL Keyword” on page 310.

BACKUP VSAM_IN Command

Use the BACKUP VSAM_IN command to create an Instant Snapshot copy of an individual VSAM file and register the copy in the repository. The keywords that you can code on this command are almost identical to the keywords that you would code to create an RUV backup of an individual file, as explained in “Coding JCL for RUV Backups of Individual VSAM Files” on page 290. However, do not code the BACKUP_OUT keyword; instead, code the VSAM_NAME_MODEL keyword to identify the VSAM name model that tells RUV how to allocate the Instant Snapshot copy, as explained in “VSAM_NAME_MODEL Keyword” on page 310.

BACKUP VSAM_GROUP Command

Use the BACKUP VSAM_GROUP command to create an Instant Snapshot copy of a defined group of VSAM files and register the copy in the repository. The keywords that you can code on this command are almost identical to the keywords that you would code to create an RUV backup of a VSAM group, as explained in “Coding JCL for RUV Backups of VSAM Groups” on page 294. However, do not code the OUTPUT_MODEL keyword; instead, code the VSAM_NAME_MODEL keyword to identify the VSAM name model that tells RUV how to allocate the Instant Snapshot copy, as explained in “VSAM_NAME_MODEL Keyword” on page 310.
COPY VSAM_IN Command

Use the COPY VSAM_IN command to create an Instant Snapshot copy of an individual VSAM file without registering the copy in the repository. The keywords that you can code on this command are almost identical to the keywords that you would code to create an RUV backup of an individual file, as explained in “Coding JCL for RUV Backups of Individual VSAM Files” on page 290. However, do not code the BACKUP_OUT keyword; instead, code the VSAM_NAME_MODEL keyword to identify the VSAM name model that tells RUV how to allocate the Instant Snapshot copy, as explained in “VSAM_NAME_MODEL Keyword” on page 310.

COPY VSAM_GROUP Command

Use the COPY VSAM_GROUP command to create an Instant Snapshot copy of a defined group of VSAM files without registering the copy in the repository. The keywords that you can code on this command are almost identical to the keywords that you would code to create an RUV backup of a VSAM group, as explained in “Coding JCL for RUV Backups of VSAM Groups” on page 294. However, do not code the OUTPUT_MODEL keyword; instead, code the VSAM_NAME_MODEL keyword to identify the VSAM name model that tells RUV how to allocate the Instant Snapshot copy, as explained in “VSAM_NAME_MODEL Keyword” on page 310.

VSAM_NAME_MODEL Keyword

Use the VSAM_NAME_MODEL keyword on the BACKUP VSAM_IN command or the BACKUP VSAM_GROUP command to create an Instant Snapshot copy and register it as a backup in the RUV repository. Use the VSAM_NAME_MODEL keyword on the COPY VSAM_IN command or the COPY VSAM_GROUP command to create an Instant Snapshot copy and do not register it as a backup in the RUV repository.

The VSAM_NAME_MODEL keyword also specifies the name of a pre-existing VSAM name model that you want RUV to use to construct the data set name of the copy. SUF controls all other characteristics of the copy. The use of the BACKUP_OUT or OUTPUT_MODEL keyword is optional unless you are making deferred backups of an Instant Snapshot copy.

Specify a single specific user-defined name (1–32 characters) as the value of the VSAM_NAME_MODEL keyword value.
Instant Snapshot Copy Examples

In the following example, the VSAM_NAME_MODEL keyword on the BACKUP VSAM_IN command tells RUV to create a copy of the file that is identified by the VSAM_IN keyword. After creating the copy, RUV registers the copy in the repository. RUV constructs the data set name of the copy by using the BACKUP1 VSAM name model.

```
SET BACKUP_METHOD(SNAPSHOT) XBMDID(XBMD); 
ADD VSAM_NAME_MODEL(BACKUP1) 
  VSAM_FILE_MASK(PAY*, 
    <NODE1>.<NODE2>.<RUVDATE>.BACKUP1); 
BACKUP VSAM_IN(PAYROLL.TAXWTHLD.TXYR2006) 
  VSAM_NAME_MODEL(BACKUP1) 
  COMMENT ('CREATE INSTANT SNAPSHOT COPY');
```

In the preceding example, the BACKUP1 VSAM name model is created in this job (with the ADD VSAM_NAME_MODEL command), but it could have been created previously. The SET command enables the use of SUF with the BACKUP_METHOD keyword and specifies use of a test XBM subsystem (XBMD) with the XBMDID keyword. These keywords could have been set as defaults previously.

RUV creates and registers this copy with the following data set name:

```
PAYROLL.TAXWTHLD.D2006292.BACKUP1
```

In the following example, the VSAM_NAME_MODEL keyword on the COPY VSAM_GROUP command tells RUV to create an Instant Snapshot copy of each data set in the PAYROLL group and construct the data set names of the copies by using the BACKUP1 VSAM name model. The copies are not registered in the repository.

```
COPY VSAM_GROUP(PAYROLL) 
  VSAM_NAME_MODEL(BACKUP1) 
  INVENTORY_ONLY(YES) 
  COMMENT ('CREATE INSTANT SNAPSHOT COPY');
```

The INVENTORY_ONLY keyword tells RUV to report the actions that it would take but not process the files. This example assumes that the BACKUP1 VSAM name model has been created previously and that previously set defaults enable SUF and specify the XBM subsystem ID.
Using the RBA0 Backup Detection Feature

The following sections describe the RBA0 backup detection feature.

Overview of the RBA0 Backup Detection Feature

Many organizations use KSDS or ESDS files for data-entry purposes. These files are typically cleared of all data by using an IDCAMS delete/define operation, followed by the load of a dummy record to the file. The file is then opened to CICS and data is entered by online operators. This process is repeated one or more times a day. There is no relationship between sets of data records other than that the records are stored under the same VSAM data set name.

RBA0 Backups

The RUV RBA0 backup detection feature registers a backup of a file at the point in time when the file was empty. It can also create an RBA0 backup when a file that has been defined with the REUSE option is opened for output. This action causes the high used relative byte address (RBA) to be reset to zero, which effectively deletes all records in the file.

An RBA0 backup is simply a record in the repository; no physical data set is associated with an RBA0 backup. However, an RBA0 backup is intended to be used, in the same way as any other type of backup, as a starting point for recovery. When an RBA0 backup is selected during restore or recovery processing, it causes a delete/define operation for the VSAM file.

Forward Recovery with RBA0 Backups

You can perform a forward recovery of a file across RBA0 backups. For more information about the actions that RUV takes during forward recovery if you are using the RBA0 backup detection feature, see “Forward Recovery and the RBA0 Backup Detection Feature” on page 452.

Identifying an RBA0 Backup

When you are viewing information about backups with the RUV ISPF interface, RBA0 backups contain information similar to the following:

```
```
Implementing the RBA0 Backup Detection Feature

To implement the RBA0 backup detection feature, use an RUV rule set to register the VSAM file and all programs that update the file:

- To register the file, code the ADD VSAM_FILE command or the ADD VSAM_GROUP_DEFINITION command.
- To register the programs that update the file, code the ADD JOB_SET command.

RUV must detect all updates against the file, especially the initial load. When a file is opened, RUV detects that the file is empty and records an RBA0 backup in the repository. A dummy record is then journaled to an RUV archive. CICS performs its own journaling operations as the file is updated. RUV detects these updates when the CICS log stream or extent is archived.

Normally, the initial load of a VSAM file is not journaled. The RUV rule set either excludes the name of the program that is performing the load or excludes the VSAM file data set name for the load job. Therefore, to enable the RBA0 detection feature, you must modify the rule set to include both the loading program name and the data set name of the VSAM file. If IDCAMS or another general-purpose utility is being used to process the file, use a job-level rule set to restrict RBA0 detection to selected files.

For complete information about constructing RUV rule sets, see Chapter 4, “Using the Batch Journaling Facility.”
Using SMS Backups and IDCAMS REPRO Copies

RUV supports the use of SMS backups and IDCAMS REPRO copies as described in the following sections.

Automatically Using SMS Backups

RUV does not invoke SMS to perform backups. If you are already creating SMS backups, you do not need to change your backup jobs to implement RUV. RUV recognizes SMS backups automatically during restore and recovery processing. For automatic handling of SMS backups, the only action that you might need to take is to code the SMS value of the BACKUP_METHOD keyword on the SET or SET DEFAULT command. The SMS value is one of the default values of the BACKUP_METHOD keyword; therefore, you do not need to code it explicitly unless this default value has been changed in your environment.

If you do not want RUV to select SMS backups, you must code BACKUP_METHOD(NO_SMS) on the SET or SET DEFAULT command.

For information about how RUV handles SMS backups during restore processing, see Chapter 10, “Restoring Data Sets.”

Automatically Using IDCAMS REPRO Copies

RUV can recognize and automatically restore copies that were created by using the REPRO command of the IDCAMS utility. For automated handling of a REPRO copy, you must register the copy as a backup file in the RUV repository. Use the REGISTER BACKUP_FILE command of the RUVZSM0 utility, as described in “Explicitly Registering Backups” on page 354.

RUV can restore the IDCAMS copy under the following conditions:

- If the VSAM file is not registered, you must name the BACKUP_IN input file as you do when the REPRO file is not registered to RUV and the VSAM file to be restored is registered to RUV.

- If the REPRO file is registered to RUV, RUV will automatically consider the REPRO file to restore your VSAM file without your specifying the file with the BACKUP_IN keyword. RUV dynamically defines the VSAM file and uses the REPRO backup for the restore operation.
Creating External Backups

The following sections describe how to use RUV to create external backups by using methods other than those that are described previously in this chapter.

Overview of External Backup Support

RUV provides a general method for support of external backups that are created by third-party software or in-house processes. This external backup support is a direct replacement for the SUBMIT function in the BMC Software RECOVERY PLUS for CICS/VSAM product. You can use external backup support to build and submit backup jobs and to register non-RUV backups so that RUV can restore them automatically during a restore or recovery process with no manual intervention.

Job JCL Records

External backup support relies on job JCL records in the repository. A job JCL record contains skeleton JCL to execute a job, such as a backup or restore process. The JCL can contain system, RUV, and user variables that RUV resolves and substitutes before submitting the job for execution. RUV supports standard JCL statements and constructions in job JCL records; in addition, RUV provides powerful action variables for processing groups and handling variables.

Creating job JCL records is typically a one-time setup process that is easy to perform with the RUV ISPF interface. You can also use the ADD JOB_JCL command of the RUVZSM0 utility.
Variables

Job JCL records support the use of variables for easy setup and maintenance. A variable is a placeholder for a value that RUV resolves during execution and substitutes at the same position as the variable. The operating system and RUV provide a predefined set of standard variables. If these variables do not meet all of your needs, you can create your own user-defined variables. You can also use any existing user-defined variables that are defined in your environment.

User-defined and RUV-provided variables are coded within angle brackets; <SYSNDX> is an example of a variable. Some RUV variables, such as <SYSNDX>, <VSAM_DEFINITION>, <VSAM_PATH_DEFINITION>, <VSAM_DSN>, and <NODEn> are valid in backup and restore job JCL records only when you code them within a set of <:BEGIN_LOOP> and <:END_LOOP> action variables.

The following RUV-defined variables are not supported for external backups:

- <BACKUP_DATE>
- <BACKUP_DSN>
- <BACKUP_TIME>
- <RUNTIME>
- <RUNTIME_HEX>

For more information, see “Using Variables” on page 173.

Action Variables

Within a job JCL record, RUV supports the use of action variables. Action variables control the substitution process rather than causing the substitution of a value.

The first character of an action variable is a colon, and action variables are coded within angle brackets; <:BEGIN_LOOP> is an example of an action variable. For more information, see “Using Action Variables” on page 179.

Submit Function

RUVD provides a Submit function that you can use for external backup support. In a two-step process, RUV generates a job by resolving the variables in the specified job JCL record (according to standard processing and any coded action variables that you have included). Then RUV can submit the job to JES for execution.

The ISPF interface supports the Submit function for backup and restore job JCL records. The RUVDZSM0 utility supports the SUBMIT BACKUP command for backup job JCL records.
Implementing External Backup Support

To implement external backup support, perform the following tasks:

1. Choose one or more VSAM files, VSAM groups, or both to select for backup.

   A RUV VSAM group identifies the VSAM files that are included in the group (and, if necessary, the files that are not included in the group). You can use existing RUV VSAM groups if you have defined them already. For more information, see Chapter 15, “Using VSAM Groups.”

2. Choose the variables to use in the job JCL record.

   If predefined RUV and system variables do not meet your needs, you can create your own variables. For more information, see “Creating User Variables” on page 171.

3. Create a job JCL record to contain the backup JCL.

   You can create the record with the ADD JOB_JCL command of the RUVZSM0 utility or with the RUV ISPF interface. The job JCL should contain a step to perform the backup and a step to register the backup. For details, see “Working with Job JCL Records for Backup and Restore Processes” on page 327.

4. Create a job JCL record to contain the restore JCL.

   You can create the record with the ADD JOB_JCL command of the RUVZSM0 utility or with the RUV ISPF interface. For details, see “Working with Job JCL Records for Backup and Restore Processes” on page 327.

5. Test the backup JCL.

   The easiest way to determine whether the backup JCL works the way that you intend is to use the RUV ISPF interface. Or you can use the Submit function of the RUVZSM0 utility and use the INVENTORY_ONLY keyword to prevent the job from being executed. For details, see “Submitting Job JCL Records for Execution” on page 334.

6. Submit the backup JCL for RUV substitution and system execution.

   You can use the Submit function of the RUVZSM0 utility, or you can use the RUV ISPF interface. The task is same as testing the backup JCL (step 5), except that you continue the process to submit the job for execution.

7. Check the results of the execution.

8. Register the restore job JCL record or the restore method.
9 Test the restore JCL.

For details, see Chapter 10, “Restoring Data Sets.”

Example of External Backup Support

The following extended example illustrates the use of RUV support for external backups.

Creation of a Group and User-Defined Variables

The following control statements define a simple VSAM group, named APY_FILES, that includes two VSAM files:

```
ADD VSAM_GROUP_DEFINITION(APY_FILES)
  VSAM_FILE(APYPROD.ACCOUNTS, INCLUDE)
  VSAM_FILE(APYPROD.DESC, INCLUDE);
```

The following control statements create user-defined variables to be used in the job JCL record:

```
ADD USER_VARIABLES(APY_SYSTEM);
<APP> /APY/
<ACCOUNT> /47318/
END_DATA;
```
Creation of a Job JCL Record to Perform Backups

Figure 54 shows example control statements to create a job JCL record to perform the backups.

Figure 54  Example ADD JOB_JCL Command for Backups

```
ADD JOB_JCL(SAMPLE_BACKUP);
//<APP>BKUP JOB (<ACCOUNT>),CLASS=A
//******* STEP 1 - PERFORM BACKUPS WITH IDCAMS
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=* 
//*** LOOP 1A
:<BEGIN_LOOP>
//B<SYSNDX> DD DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,
//          DSN=<NODE1>.<NODE2>.BACKUP(+1)
:<END_LOOP>
//*** LOOP 1B
//SYSIN DD *
:<BEGIN_LOOP>
REPRO INDATASET(<VSAM_DSN>) -
    OUTFILE(B<SYSNDX>)
:<END_LOOP>
//******* STEP 2 - REGISTER BACKUPS WITH RUV
//STEP2 EXEC PGM=RUVZSM0
//SYSPRINT DD SYSOUT=* 
//*** LOOP 2A
:<BEGIN_LOOP>
//B<SYSNDX> DD DISP=SHR,DSN=*.STEP001.B<SYSNDX>
:<END_LOOP>
//*** LOOP 2B
//SYSIN DD *
:<BEGIN_LOOP>
REGISTER BACKUP_FILE(B<SYSNDX>)
    VSAM_FILE(<VSAM_DSN>)
    LOCATION(LOCAL);
:<END_LOOP>
END_DATA;
```

The JCL in this job JCL record performs an IDCAMS REPRO copy of each member of a group of VSAM files and registers the resulting external backups in the RUV repository.

- The job JCL record is named SAMPLE_BACKUP.
- In the JOB statement, the user-defined variables <APP> and <ACCOUNT> represent the application identifier and the account number.
The `<:BEGIN_LOOP>` action variable identifies the beginning of a set of statements to be repeated for each member of the VSAM group. The `<:END_LOOP>` action variable identifies the end of a set of statements to be repeated for each member of the VSAM group. The set of statements between the `<:BEGIN_LOOP>` and `<:END_LOOP>` action variables is called a loop.

The `<SYSNDX>` variable represents a seven-digit counter that RUV initializes to 0000001 when it encounters the `<:BEGIN_LOOP>` action variable and increments by 1 when it repeats the loop for another member of the VSAM group. Note that the `<SYSNDX>`, `<VSAM_DEFINITION>`, `<VSAM_PATH_DEFINITION>`, `<VSAM_DSN>`, and `<NODE>` RUV variables are valid only within a set of `<:BEGIN_LOOP>` and `<:END_LOOP>` action variables.

Within Step 1 (which executes the IDCAMS utility), Loop 1A allocates the output backup data set for a member of the VSAM group. Loop 1B performs the backup of a member of the VSAM group.

Within Step 2 (which executes the RUVZSM0 utility), Loop 2A allocates the backup data set that was created in the first job step. Loop 2B registers the backup data set in the repository.

**Creation of a Job JCL Record to Perform Restores**

Figure 54 shows example control statements to create a job JCL record to perform the restores.

**Figure 55  Example ADD JOB_JCL Command for Restores (Part 1 of 2)**

```
ADD JOB_JCL(SAMPLE_RESTORE);
//<APP>RECOV JOB (<ACCOUNT>)CLASS=A
****** STEP 1  - PERFORM RECOVERY WITH IDCAMS
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*  
//SYSSIN DD *  
//:BEGIN_LOOP
DEL <VSAM_DSN> PURGE
SET MAXCC = 0
//:END_LOOP
/**
//STEP2 EXEC PGM=IDCAMS
//SYSSOUT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSSIN DD *  
//:BEGIN_LOOP
<VSAM_DEFINITION>
<VSAM_PATH_DEFINITION>
//:END_LOOP>
```
The first job step performs an IDCAMS delete operation for the files to be restored. The second step performs the IDCAMS define operation for the files; the file definitions are obtained from the VSAM file records in the RUV repository. The third and subsequent steps perform the restore operations from the external backups.

Submit Backup Command

The backup process starts with a SUBMIT BACKUP command:

```plaintext
SET INTERNAL_READER(JES);
SUBMIT BACKUP JOB_JCL(SAMPLE_BACKUP)
  VSAM_GROUP(APY_FILES);
```

This command specifies the use of the job JCL record that is named SAMPLE_BACKUP to create external backups of the members of the VSAM group named APY_FILES.
**Generated Backup JCL**

In response to the SUBMIT BACKUP command, RUV performs variable substitution and generates the JCL that is shown in Figure 56.

**Figure 56  Example Generated Backup JCL**

```plaintext
//APYBKUP   JOB (47318),CLASS=A
/**********  STEP 1 - PERFORM BACKUPS WITH IDCAMS
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=* 
/******** LOOP 1A
//B0000001  DD DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,
//          DSN=APYPROD.ACCOUNTS.BACKUP(+1)
//B0000002  DD DISP=(NEW,CATLG,DELETE),
//          UNIT=SYSDA,
//          DSN=APYPROD.DESC.BACKUP(+1)
/******** LOOP 1B
//SYIN DD * 
  REPRO INDATASET(APYPROD.ACCOUNTS) -
     OUTFILE(B0000001)
  REPRO INDATASET(APYPROD.DESC) -
     OUTFILE(B0000002)
/**********  STEP 2 - REGISTER BACKUPS WITH RUV
//STEP2     EXEC PGM=RUVZSM0
//SYSPRINT  DD SYSOUT=* 
/******** LOOP 2A
//B0000001  DD DISP=SHR,DSN=*.STEP001.B0000001
//B0000002  DD DISP=SHR,DSN=*.STEP001.B0000002
/******** LOOP 2B
//SYIN DD *
  REGISTER BACKUP_FILE(B00000001)
     VSAM_FILE APYPROD.ACCOUNTS )
     LOCATION(LOCAL);
  REGISTER BACKUP_FILE(B00000002)
     VSAM_FILE APYPROD.DESC   )
     LOCATION(LOCAL);
```

Because the INVENTORY_ONLY(YES) keyword is not included in the job JCL record, RUV submits the generated JCL to JES for execution.

**NOTE**

No changes should be allowed to occur to the file from the beginning of STEP1 to the end of STEP2.
**Restore Command**

During the recovery process, RUV can automatically restore a file that was backed up by using the SUBMIT BACKUP JOB_JCL command (if the backup was registered). For an automatic restore, RUV forms all VSAM files that have the same RESTORE JOB_JCL name into a group, performs variable substitutions, builds the RESTORE JOB_JCL command internally, and submits the restore JCL for execution.

When RUV submits an automated restore, RUV needs to know when the submitted job has completed so that RUV can proceed to apply archive records. RUV uses an external job record to track the status of the restore job. RUV creates and deletes external job records automatically; you do not need to handle them unless you cancel the restore job. For more information, see “Working with External Job Records” on page 343.

You can execute the RESTORE VSAM_FILE or RESTORE VSAM_GROUP command if you manually want to restore a backup without applying journal updates at that time.

The following RESTORE VSAM_FILE command causes RUV to use the restore job JCL record to perform the restore of the APYPROD.ACCOUNTS file:

```
//jobcard
//RESTORE EXEC PGM=RUVZSM0,REGION=64M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   SET SUBSYSTEM(ssid);
   SET REPORT(DETAIL);
   RESTORE VSAM_FILE(APYPROD.ACCOUNTS);
/*
```
Generated Restore JCL

When RUV generates the RESTORE command from the restore job JCL record, RUV performs variable substitution and generates the JCL that is shown in Figure 56.

Figure 57  Example Generated Restore JCL (Part 1 of 3)

```plaintext
//APYRECOV  JOB (47318),CLASS=A
//******* STEP 1  - PERFORM RECOVERY WITH IDCAMS
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*  
//SYSIN     DD * 
DEL APYPROD.ACCOUNTS
SET MAXCC = 0
DEL APYPROD.DES
SET MAXCC = 0

//STEP2 EXEC PGM=IDCAMS
//SYSOUT    DD SYSOUT=*  
//SYSPRINT  DD SYSOUT=* 
//SYSIN     DD * 
/* DEFINE CLUSTER */
DEFINE CLUSTER ( -
               NAME( APYPROD.ACCOUNTS               ) -
               INDEXED -
               KEYS(000000006 00000000) -
               RECSZ(00001000 00001000) -
               ) -
               DATA ( -
               NAME( APYPROD.ACCOUNTS.DATA          ) -
               CYL(000000010 000000005) -
               VOL(DEV528 -
               ) -
               BUFSP(00009728) -
               CISZ(00004096) -
               NOERASE -
               KEYS(000000006 00000000) -
               RECSZ(00001000 00001000) -
               NOREUSE -
               SHR( 2  3 ) -
               NISPND -
               RECOVERY -
               NWCK -
               ) -
               INDEX ( -
               NAME( APYPROD.ACCOUNTS.INDEX          ) -
               TRK(000000001 000000001) -
               VOL(DEV528 -
               ) -
               CISZ(00001536) -
               NOIMBED -
               NREPL -
               NOREUSE -
               SHR( 2  3 ) -
               NWCK -
               )
```
/* no PATH found for APYPROD.ACCOUNTS */
DEFINE CLUSTER ( -
    NAME(APYPROD.DESC ) -
    FSPC(010 010) -
    INDEXED -
    KEYS(00000014 00000004) -
    RECSZ(00000100 00000500) -
) -
DATA ( -
    NAME(APYPROD.DESC.DATA ) -
    TRK(00000001 00000001) -
    VOL(EM565A -
    ) -
    BUFSP(00008704) -
    CISZ(00004096) -
    NOERASE -
    KEYS(00000014 00000004) -
    RECSZ(00000100 00000500) -
    NOREUSE -
    SHR( 2 3 ) -
    NSPND -
    RECOVERY -
    NWCK -
) -
INDEX ( -
    NAME(APYPROD.DESC.INDEX ) -
    TRK(00000001 00000001) -
    VOL(EM565A -
    ) -
    CISZ(00000512) -
    NOIMBED -
    NREPL -
    NOREUSE -
    SHR( 2 3 ) -
    NWCK -
)
Register the Restore Job JCL or Restore Method

The following example shows the registration of a non-RUV backup in the repository by using the REGISTER RESTORE_JOB_JCL command:

```plaintext
REGISTER RESTORE_JOB_JCL(RESTORE1)  
  VSAM_FILE(APYPROD.ACCOUNTS)  
  START_TIME(20093200000000)  
  PARM(Y)  
  LOCATION(LOCAL)  
  COMMENT('RESTORE ACCOUNTS FILE')  
  STATUS(ACTIVE)
```

The following example shows the registration of a non-RUV backup in the repository by using the REGISTER RESTORE_METHOD command:

```plaintext
REGISTER RESTORE_METHOD(RESTORE2)  
  VSAM_FILE(APYPROD.ACCOUNTS)  
  START_TIME(20093200000000)  
  PARM(Y)  
  LOCATION(LOCAL)  
  COMMENT('WTOR TO RESTORE ACCOUNTS')  
  STATUS(ACTIVE)
```
The following sections describe how to work with job JCL records, also called skeleton JCL, to execute external backup and restore processes.

RUV also uses job JCL records to contain skeleton JCL for performing job step backout. For information about working with job JCL records for this purpose, see “Working with Job JCL Records for Backout” on page 182.

Overview of Job JCL Records

External backup support relies on job JCL records in the repository. A job JCL record contains skeleton JCL to execute a job, such as a backup or restore process. The JCL can contain system, RUV, and user variables that RUV resolves and substitutes before submitting the job for execution. RUV supports standard JCL statements and constructions in job JCL records; in addition, RUV provides powerful action variables for processing groups and handling variables.

When the job JCL record is ready to use, you can use the RUV Submit function to submit the job JCL record for execution. For more information, see “Submitting Job JCL Records for Execution” on page 334.

You can work with job JCL records by using the ISPF interface or the RUVZSM0 utility. You can create, update, and delete job JCL records, and you can display the contents of job JCL records.

Creating a Job JCL Record with the ISPF Interface

For information about using the ISPF interface to create a job JCL record, see “Using the ISPF Interface to Work with Rule Sets” on page 205. The following steps summarize the process:

1. Access the ISPF interface as described in “Using the RUV ISPF Interface” on page 113.

   RUV displays the RUV Options panel (Figure 19 on page 115).

2. Select option 8 (Subsystem / Ruleset Information).

   RUV displays the Subsystem Detail panel (Figure 36 on page 205).
Enter any character in the choice entry field to the left of **Rule Set**.

RUV displays the Ruleset List panel (Figure 37 on page 206).

If you do not want to display a list of existing job JCL records, go to step 6. If you want to display a list of existing job JCL records, on the Ruleset List panel enter any character in the choice entry field to the right of **Show**.

RUV displays the Show pop-up window (Figure 38 on page 207). Or

In the Show pop-up window, enter 5 in the choice entry field.

RUV displays the Ruleset List window with only job JCL records displayed.

On the Ruleset List panel, select the **File** menu, and then select option 1 (New).

RUV displays the RUV Ruleset/Group/Model Templates panel (Figure 43 on page 211).

In the **Option** field, enter R4 to create a job JCL record.

RUV displays an ISPF Edit panel that contains model statements for a job JCL record (Figure 58).

---

**Figure 58  ISPF Edit Panel for Job JCL**

---

```plaintext
* Enter your JOB_JCL name
ADD JOB_JCL(your name)
COMMENT("Sample comment");

/*JOBNAME(1,8) JOB (<ACCOUNT>),CLASS=A,MSGCLASS=X,REGION=3M
PRERECOVERY STEPS AS REQUIRED BY USER
ZSM0 EXEC PGM=RUVZSM0 <- RUV Recovery Step
STEPLIB DD if required
SYSPRINT DD SYSOUT=* 
SYSSIN DD *
<RUV_CONTROL>

/* POSTRECOVERY STEPS AS REQUIRED BY USER
END_DATA;

=NOTE= Syntax: JCL format with variable substitution
=NOTE= The JCL may be used to build a recovery job stream in the event
=NOTE= of a user abend. This jobstream will perform step back out.
=NOTE= For a step backout, <RUV_CONTROL>, <RUV_JOURNAL_DEACTIVATE>,
and <RUV_STEPLIB> are valid user variables.
```
8 Edit the job JCL as needed. You must provide a unique name for the ADD JOB_JCL command.

9 When you have made all of the necessary changes, enter the END command.

RUV saves the job JCL record with the name that you provided for the ADD JOB_JCL command in the JCL.

Creating a Job JCL Record with the RUVZSM0 Utility

To use the RUVZSM0 utility to create a job JCL record, perform the following steps:

1 Code the JCL to execute the ADD JOB_JCL command of the RUVZSM0 utility.

   For details, see “Coding JCL to Work with Job JCL Records” on page 329.

2 Submit the JCL for execution.

3 Check the results of the execution.

Coding JCL to Work with Job JCL Records

Figure 50 shows model JCL for creating a job JCL record for a backup or restore job with the ADD JOB_JCL command.

**Figure 59 ADD JOB_JCL Model JCL for a Backup Job**

```plaintext
//jobname JOB job statement
//ADJOBJCL EXEC PGM=RUVZSM0
//STEPLIB DD DISP=SHR,DSN=yourname.LOAD
//SYSPRINT DD SYSOUT=*

ADD JOB_JCL(name32)
   | COMMENT(comment_fields) |
* skeleton JCL statements can include RUV variables, system
* variables, user-defined variables, and action variables
END_DATA;

//*
```
ADD JOB_JCL Command

Use the ADD JOB_JCL command of the RUVZSM0 utility to create a job JCL record.

The value of the JOB_JCL keyword is the 1- to 32-character name that you want to use for the job JCL record.

Skeleton JCL Statements and Variables

Skeleton JCL statements are similar to the standard statements that you would code for a backup or restore job. However, you can include the following types of elements in the statements:

- RUV variables (as described in “Using RUV-Defined Variables” on page 174)
- system variables (as described in “Using IBM Variable Substitution Symbols” on page 178)
- user-defined variables (as described in “Creating User Variables” on page 171)
- action variables (as described in “Using Action Variables” on page 179)

END_DATA Keyword

Use the END_DATA keyword to indicate the end of the skeleton JCL.

UPDATE JOB_JCL Command

Use the UPDATE JOB_JCL command to update an existing job JCL record in the repository. Use the following syntax:

```
UPDATE JOB_JCL(name32)
  { COMMENT(comment_fields) }
* skeleton JCL statements can include RUV variables, system
* variables, user-defined variables, and action variables
END_DATA;
```

This command is identical to the ADD JOB_JCL command.
DELETE JOB_JCL Command

Use the DELETE JOB_JCL command to delete a job JCL record from the repository. Use the following syntax:

```
DELETE JOB_JCL(name32);
```

REPORT JOB_JCL Command

Use the REPORT JOB_JCL command to create a report that displays the contents of a job JCL record. Use the following syntax:

```
REPORT JOB_JCL(name32);
```

Example Job JCL Records

For an example of a job JCL record for backup processing, see “Creation of a Job JCL Record to Perform Backups” on page 319. For an example of a job JCL record for restore processing, see “Creation of a Job JCL Record to Perform Restores” on page 320.
Storing Information into a File

You might want to record information, such as the current time, in a different location than the job JCL record so that it is easy to retrieve for subsequent use. You can use the STORE INTO_FILE command of the RUVZSM0 utility for this purpose.

Coding the STORE INTO_FILE Command

To code the STORE INTO_FILE command, use the following syntax:

```
STORE INTO_FILE ( dsn44_or_pdsmember | <variables> ) ;
text with <variables>
END_DATA;
```

When the command is executed, RUV performs variable substitution on the set of statements between the STORE INTO_FILE command and the END_DATA marker.

INTO_FILE Keyword

RUV writes the results of the STORE INTO_FILE command to the sequential data set or partitioned data set member that is identified by the INTO_FILE keyword. You can specify the data set name (and member name) explicitly, or you can build the name with variables.

Action Variables

RUV supports the use of the <:IMBED> action variable with the STORE INTO_FILE command; however, the <:BEGIN_LOOP> and <:END_LOOP> action variables are not supported with this command.

Output Records

The output of the STORE INTO_FILE command consists of fixed-length, 80-byte records. RUV creates the file or member with the NEW parameter (not the MOD parameter) each time it is opened with the STORE INTO_FILE command.

The simplest way to read this file or member in an RUV job is as a SYSIN or concatenated SYSIN file.
Example of STORE INTO_FILE Command Usage

The following example shows a typical use for the STORE INTO_FILE command. When you explicitly register an external backup, you might also want to register an accurate start time for that backup (rather than use the default time that the registration was performed).

You can use the STORE INTO_FILE command in the job step that submits the backup job JCL.

```plaintext
... 
//STEP010 EXEC PGM=RUVRZSMO 
//SYSPRINT DD SYSOUT=X 
//PDSMEM DD DISP=SHR, DSN=RUVRUVV2.EXTNBKUP.PDS(STOREV) 
//SYSIN DD * 
STORE INTO_FILE(PDSMEM); START_TIME(<RUVDATETIME(2)>) END_DATA; 
... 
```

The following data is stored in the PDS member:

START_TIME(2009329/1946320)

Then, in the REGISTER BACKUP_FILE step of the backup job JCL record, you can retrieve the stored time and use it as the value of the START_TIME keyword:

```plaintext
... 
//RGSTBKUP EXEC RUVRZSMO 
//SYSIN DD * 
    REGISTER BACKUP_FILE(SAMPLE_BACKUP) 
    DD DISP=SHR, DSN=RUVRUVV2.EXTNBKUP.PDS(STOREV) 
    DD * : 
... 
```
Submitting Job JCL Records for Execution

The following sections describe how to use the RUV Submit function to submit job JCL records for execution.

Overview of the Submit Function

When a job JCL record is ready to use, you can use the RUV Submit function to submit the job JCL record for execution. When the record is submitted, RUV performs variable substitution and generates the JCL. If INVENTORY_ONLY(NO) is specified, RUV then submits the JCL for execution.

For backup job JCL records, you can perform the RUV Submit function by using the ISPF interface or the RUZSM0 utility. For restore job JCL records, you can perform the RUV Submit function by using the ISPF interface.

Submitting a Job JCL Record with the ISPF Interface

To use the ISPF interface to submit a job JCL record, perform the following steps:

1. Access the ISPF interface as described in “Using the RUV ISPF Interface” on page 113.

   RUV displays the RUV options panel (Figure 19 on page 115).

2. Select option 10 (External Backup / Restore JCL Generation).

   RUV displays the VSAM Sphere External Backup / Restore panel (Figure 60). Use this panel to specify the required information for submitting the external backup or restore job JCL for variable substitution, JCL generation, and execution.
If the job JCL record contains user variables to be resolved, specify a rule set name in the **System level rule set** or **Job level rule set** field.

The name of the active system-level rule set is displayed by default. You can blank out the value in the field and type another value. If the value contains one or more masking characters (* or ?), RUV displays the VSAM Sphere Backup / Restore - Rule_Set Names panel (Figure 61), which lists the existing rule sets that match the mask.
4 Select the rule set that you want to use by typing any character to the left of the rule set name and pressing Enter, and then enter the END command to return to the VSAM Sphere Backup / Restore panel.

5 In the VSAM file mask or VSAM group mask field, enter the fully qualified name of the VSAM file or VSAM group that you want to select, or enter a value with masking characters (*) and (?) to view a list of existing VSAM files or VSAM groups.

If you entered a masked value, RUV displays the VSAM Sphere Backup / Restore - VSAM File Name panel (Figure 62) or the VSAM Sphere Backup / Restore - VSAM Group Name panel (not shown).

If you want RUV to display the VSAM Sphere Backup / Restore - VSAM File Name panel or the VSAM Sphere Backup / Restore - VSAM Group Name panel with all files or groups already selected, enter any character in the Preselect all files or groups field. You can then remove the selection for the files or groups that you do not want to include in the command by entering any character to the left of the file or group name.
6 Select one or more files or groups by entering any character to the left of the file or group name. Then enter the END command to return to the VSAM Sphere Backup / Restore panel.

7 In the Job JCL mask field, enter the name of the existing job JCL record that you want to use.

If you enter a masked value that contains one or more masking characters (* or ?), RUV displays the VSAM Sphere Backup / Restore - Rule_Set Names panel (Figure 63), which lists the existing job JCL records that match the mask.
Select the job JCL record that you want to use by typing any character to the left of the job JCL record name and pressing Enter, and then enter the END command.

An ISPF Edit panel is displayed (Figure 64). It displays a temporary data set that contains the skeleton JCL from the selected job JCL record.
You can change the displayed skeleton JCL. When you are ready for RUV to perform variable substitution, enter the PERSUB command.

RUV performs variable substitution and displays the generated JCL (Figure 65 on page 340).
10 Review the generated JCL. To make changes to the generated JCL, you can edit the generated JCL directly, but the job JCL record will not be affected. To change the job JCL record, enter the END command, and the temporary data set that contains the skeleton JCL is redisplayed so that you can change it.

You can repeat the process of entering the PERSUB command, reviewing the generated JCL, entering the END command, and changing the skeleton JCL until it produces the results you want. To save the changes that you made to the skeleton JCL and update the job JCL record in the repository, enter the END command.

**NOTE**

You must enter the END command (when the skeleton JCL is displayed) if you want to save the updated job JCL record to the repository. Entering the SAVE command does not cause the updated job JCL record to be written to the repository, and the generated JCL is never saved to the repository.

You can save the generated JCL in a permanent data set, submit the JCL for execution (with the SUBMIT command), enter the END command to return to editing the job JCL record, or enter the CANCEL command to discard the generated JCL and all changes to the job JCL record.
Submitting a Job JCL Record with the RUVZSM0 Utility

To use the RUVZSM0 utility to submit a backup job JCL record for execution, perform the following steps:

1. Choose an existing backup job JCL record to submit. The name of the job JCL record is the value that you must code for the JOB_JCL keyword on the SUBMIT command.

   For more information about job JCL records, see “Working with Job JCL Records for Backup and Restore Processes” on page 327.

2. Code the JCL to execute the SUBMIT command.

   For details, see “Coding JCL to Submit Job JCL Records for Execution” on page 341.

3. Submit the JCL for execution.

4. Check the results of the execution.

Coding JCL to Submit Job JCL Records for Execution

Figure 66 shows model JCL for submitting a backup job JCL record with the SUBMIT command.

**Figure 66  SUBMIT BACKUP Model JCL for a Backup Job**

```
//jobname JOB job statement
//SBJOBJCL EXEC PGM=RUVZSM0
//STEPLIB DD DISP=SHR,DSN=yourname.LOAD
//SYSPRINT DD SYSOUT=*  
/*
SET INTERNAL_READER(name32) ;
SUBMIT BACKUP
   JOB_JCL(name32)
      { VSAM_GROUP(name32, ...) | VSAM_FILE(ddn_or_dsn) }
      { INVENTORY_ONLY(NO | YES)};
/*
```

SUBMIT BACKUP Command

Use the SUBMIT BACKUP command of the RUVZSM0 utility to submit a backup job JCL record for execution. The required BACKUP keyword tells RUV to perform variable substitutions for a backup job.
Coding JCL to Submit Job JCL Records for Execution

JOB_JCL Keyword

Use the required JOB_JCL keyword to identify the job JCL record to be submitted for execution. The value of the JOB_JCL keyword is the 1- to 32-character name of an existing job JCL record.

VSAM_GROUP Keyword

You can use the optional VSAM_GROUP keyword to specify one or more VSAM group definitions. Each keyword value is the 1- to 32-character name of an existing VSAM group; you can code multiple values. During the substitution process, the VSAM files in the group become the source for the <VSAM_DSN> variable substitutions. According to the group definition, files are included and excluded from the group and duplicates are eliminated. If you code the VSAM_GROUP keyword, do not code the VSAM_FILE keyword.

VSAM_FILE Keyword

You can use the optional VSAM_FILE keyword to specify a VSAM file. The keyword value is the data set name of the VSAM file or the ddname of the DD statement that describes the VSAM file in the SUBMIT job JCL. During the substitution process, this VSAM file becomes the source for the <VSAM_DSN> variable substitutions. If you code the VSAM_FILE keyword, do not code the VSAM_GROUP keyword.

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword to obtain information about the files that would be processed by the function without actually performing the function. Set one of the following values. The default value is NO.

NO Perform the function.

YES Produce a listing of VSAM files that would be selected by the function and the name of each data set that would be created by the function. Do not perform the function.

Example SUBMIT BACKUP Command

For an example of a SUBMIT BACKUP command, see “Submit Backup Command” on page 321.
Working with External Job Records

When RUV performs a restore process for an external backup, RUV creates an external job record to track the progress of the restore job. In most cases, RUV automatically deletes the external job record when the record is no longer needed to track the restore job. However, if the restore job is cancelled, RUV does not have a chance to delete the record before termination.

You can execute the REPORT EXTERNAL_JOB(*) command to display all existing job JCL records and identify the keys of the records that are no longer needed. Then you can use the DELETE EXTERNAL_JOB(key) command to delete these records.

Coding REPORT and DELETE EXTERNAL_JOB Commands

To execute REPORT and DELETE EXTERNAL_JOB commands, use the syntax shown in Figure 67.

**Figure 67  REPORT and DELETE EXTERNAL_JOB Command Syntax**

<table>
<thead>
<tr>
<th>REPORT</th>
<th>DELETE EXTERNAL_JOB(key, *, ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FILTER_BY(</td>
</tr>
<tr>
<td></td>
<td>VSAM_FILE(dsn_or_ddname)</td>
</tr>
<tr>
<td></td>
<td>VSAM_GROUP(name32_or_mask)</td>
</tr>
<tr>
<td></td>
<td>SMFID(value)</td>
</tr>
<tr>
<td></td>
<td>START_TIME/yyyyjjjhhmmsst</td>
</tr>
<tr>
<td></td>
<td>STOP_TIME/yyyyjjjhhmmsst</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
</tbody>
</table>

**EXTERNAL_JOB Keyword**

The value of the EXTERNAL_JOB keyword is the key of the record, which consists of the 32-character name of the job JCL record and an appended 7-character job number that RUV assigns at execution time. You can specify a mask for the key.

**FILTER_BY Keyword**

The standard FILTER_BY keyword is valid on REPORT and DELETE EXTERNAL_JOB commands. For more information, see “Using the FILTER_BY Keyword” on page 108.
EXTERNAL_JOB Report

Figure 68 shows an example of the report that is produced with the REPORT EXTERNAL_JOB command.

Figure 68 EXTERNAL_JOB Report

REPORT EXTERNAL_JOB(*);

Restore External Job Record: EXT002B_RESTORE       V000074

Job RUV$REST JOB15749 is still Running.

VSAM Files for this Restore
   VSAM DSN: RUVRUV.QA.EXT.RUVEBASE.
   VSAM DSN: RUVRUV.QA.EXT.RUVE0010.
   VSAM DSN: RUVRUV.QA.EXT.RUVKBASE.
   VSAM DSN: RUVRUV.QA.EXT.RUVK0010.
   VSAM DSN: RUVRUV.QA.EXT.RUVR0010.

Submit Job        : RUV$TSTB  JES: JOB15741  Userid: HLK
   Stepname: RESTORE        Pgm: RUVZSMO  XCF  : SYSP     SMFID: SYSP

Restore Job       : RUV$REST  JES: JOB15749  Userid: HLK
   Stepname:         Pgm:    XCF  :    SMFID:

   Internal Reader: EXT002B_INTRDR_JES
   Job JCL        : EXT002B_RESTORE
RUVR202140I RUV EOJ. Maximum CC was 0 .
Creating Copies of Backups

You can use the Copy Backup function to copy an existing backup data set and to create a new backup data set as a QSAM data set (flat file) on tape or DASD.

Overview of the Copy Backup Function

The Copy Backup function includes the following features:

- The input backup data set can be an Instant Snapshot copy or any type of RUV backup except for an RBA0 backup.

- The function can copy backup data sets for a VSAM group (as defined with the ADD VSAM_GROUP_DEFINITION command) or for a single VSAM file.

- After you have taken an Instant Snapshot copy and immediately made the original VSAM data set available for other processing, the function allows you to create a traditional backup of the Instant Snapshot copy at your convenience. With this process, you can have the minimal data outage of an Instant Snapshot copy with the flexibility of a traditional backup.

- When the function has created an output (copied) backup data set, it can retain or scratch the input backup data set. You can use this feature to remove Instant Snapshot copies automatically from DASD after the copy is written to tape.

- You can schedule the function to be executed at any time after the original backup is complete.

- The function works with stacked tapes for input and output (if stacked tape specifications have been made).

- By default, the function selects and copies the most recent active VSAM or QSAM backups. For a VSAM file, the function can copy any active backup that you specify. For a VSAM group or a VSAM file, the function can copy the first active backup that has occurred after a timestamp that you specify. Note that this processing is different from other functions that use the latest backup.
As shown in Figure 69, use the COPY BACKUP command to create one or more copies of each selected backup. The following example JCL shows how to execute this command.

**Figure 69  COPY BACKUP Command Example JCL**

```plaintext
//*      DEFERRED COPY BACKUP
//COPYBK  EXEC PGM=RUVZSMO
//STEPLIB  DD DISP=SHR,DSN=yourname.LOAD
//SYSPRINT DD SYSOUT=*  
//D0030   DD DISP=SHR,DSN=RUVRUV.QA.DCB1A.RUVK0030
//BK3OUT  DD DISP=(,CATLG,DELETE),DSN=RUVRUV.QA.DCB1ABK.RUVK30,  
//        UNIT=SYSDA,SPACE=(TRK,(5,1,RLSE)
//BK2OUT  DD DISP=(,CATLG,DELETE),DSN=RUVRUV.QA.DCB1ABK.RUVK20,  
//        UNIT=SYSDA,SPACE=(TRK,(5,1,RLSE)
//BK1OUT  DD DISP=(,CATLG,DELETE),DSN=RUVRUV.QA.DCB1ABK.RUVK10,  
//        UNIT=SYSDA,SPACE=(TRK,(5,1,RLSE)
//SYSOUT  DD SYSOUT=*  
//SYSIN   DD * SET REPORT(DETAIL);
SET SUBSYSTEM(VDCB):
COPY BACKUP VSAM_IN (RUVRUV.QA.DCB1A.RUVK0010)  
     INVENTORY_ONLY(NO) VSAM_ONLY(YES)  
     BACKUP_IN(CURRENT_BACKUP)  
     BACKUP_OUT(BK1OUT):
COPY BACKUP VSAM_IN (RUVRUV.QA.DCB1A.RUVK0020)  
     INVENTORY_ONLY(NO) VSAM_ONLY(NO)  
     BACKUP_IN(CURRENT_BACKUP)  
     BACKUP_OUT(BK2OUT):
COPY BACKUP VSAM_IN (D0030)  
     VSAM_ONLY(NO)  
     BACKUP_IN(CURRENT_BACKUP)  
     BACKUP_OUT(BK3OUT):
```
COPY BACKUP Command for a VSAM Group

To execute the Copy Backup function for a VSAM group, use the syntax shown in Figure 70.

Figure 70 COPY BACKUP Command Syntax for a VSAM Group

```
COPY BACKUP VSAM_GROUP(name32)
  {INVENTORY_ONLY(NO|YES)}
  {DELETE(NO|YES)}
  {SCRATCH(NO|YES)}
  {VSAM_ONLY(YES|NO)}
  {START_TIME(yyyydddhhmmsst|Xhhhhhhhhhhhhhhhh)}
  {PREFIX(YES|NO)}
OUTPUT_MODEL(name32)
  {COMMENT('comment_fields')}
  {LOCATION(location_name8)}
  {STATUS(ACTIVE|INACTIVE)}
)
{OUTPUT_MODEL(...)}
;
```

COPY BACKUP Command for a VSAM File

To execute the Copy Backup function for a VSAM file, use the syntax shown in Figure 71.

Figure 71 COPY BACKUP Command Syntax for a VSAM File

```
COPY BACKUP VSAM_FILE(dsn44)
  {INVENTORY_ONLY(NO|YES)}
  {DELETE(NO|YES)}
  {SCRATCH(NO|YES)}
  {VSAM_ONLY(YES|NO)}
  {BACKUP_IN(CURRENT_BACKUP | ddn8 | dsn44)}
  {START_TIME(yyyydddhhmmsst|Xhhhhhhhhhhhhhhhh)}
  {PREFIX(YES|NO)}
BACKUP_OUT(ddn8)
  {COMMENT('comment_fields')}
  {LOCATION(location_name8)}
  {STATUS(ACTIVE|INACTIVE)}
)
{BACKUP_OUT(...)}
;
```

VSAM_GROUP Keyword

To perform the Copy Backup function for a defined group of VSAM files, specify the VSAM_GROUP keyword. (A VSAM group is defined with the ADD VSAM_GROUP_DEFINITION command.) Specify the keyword value as the name (1–32 characters) of the VSAM group that you want the function to select.
**VSAM_FILE Keyword**

To perform the Copy Backup function for a single VSAM file, specify the VSAM_FILE keyword. Specify the keyword value as the fully qualified ddname (1–8 characters) or data set name (1–44 characters) of the VSAM file that you want to select. Masking characters are not accepted.

**INVENTORY_ONLY Keyword**

You can use the optional INVENTORY_ONLY keyword to obtain information about the files that would be processed by the function without actually performing the function. Set one of the following values. The default value is **NO**.

- **NO** Perform the function.
- **YES** Produce a listing of VSAM files that would be selected by the function and the name of each copy of a backup that would be created by the function. Do not perform the function.

**DELETE Keyword and SCRATCH Keyword**

You can use the optional DELETE keyword and SCRATCH keyword to control how the Copy Backup function handles the original backup data set and the repository record that contains information about this backup. These keywords work as shown in the following matrix:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>SCRATCH(NO)</th>
<th>SCRATCH(YES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE(NO)</td>
<td>The original backup data set is left as it is, and the backup registration record is left in repository.</td>
<td>Return code 8 is issued, and no action is taken against the original backup data set or the backup registration record in the repository.</td>
</tr>
<tr>
<td>DELETE(YES)</td>
<td>The original backup data set is left as it is, and the backup registration record is removed from the repository.</td>
<td>The original backup data set is scratched, and the backup registration record is removed from the repository.</td>
</tr>
</tbody>
</table>
DELETE Keyword

Use the DELETE keyword to control whether the function deletes the backup registration record from the repository after the copy is complete. Set one of the following values. The default value is NO.

NO  Leave the backup registration record in the repository.

YES  Delete the backup registration record from the repository.

SCRATCH Keyword

You can use the optional SCRATCH keyword to control whether the original backup data set is scratched (deleted) after the copy is complete. The primary purpose of this keyword is to reclaim the space that an Instant Snapshot copy occupies on the storage device after the backup is copied to tape. Set one of the following values. The default value is NO.

NO  Leave the original backup data set as it is.

YES  Scratch the original backup data set after the copy is complete.

VSAM_ONLY Keyword

You can use the optional VSAM_ONLY keyword to control the type of backups that the Copy Backup function selects to copy. Set one of the following values. The default value is YES.

YES  Select Instant Snapshot copies only.

NO  Select any type of RUV backup (except an RBA0 backup).

--- NOTE ---
The BACKUP_METHOD keyword on the SET or SET DEFAULT command can affect the type of backup that is selected.

BACKUP_IN Keyword

For a VSAM file (as specified with the VSAM_FILE keyword), by default the function selects and copies the most recent backup, but it can copy any backup. You can use the optional BACKUP_IN keyword to identify a specific active backup that you want to copy. You cannot specify BACKUP_IN(CURRENT_BACKUP) if you specify the START_TIME keyword.
For a VSAM group (as specified with the VSAM_GROUP keyword), the BACKUP_IN keyword is invalid.

Specify one of the following values. The default value is CURRENT_BACKUP.

**CURRENT_BACKUP**
Select the most current active backup.

**ddn8** or **dsn44**
Select the active backup that has the specified ddname or data set name.

### START_TIME Keyword

By default, the Copy Backup function selects and copies the most recent VSAM or QSAM backups. You can use the optional START_TIME keyword to select previous backups. If you specify the START_TIME keyword, the function selects the first backup with the following characteristics:

- It matches the type of backup that is specified with the VSAM_ONLY keyword.
- It has an active status.
- It has a timestamp that is equal to or greater than the value of the START_TIME keyword.

On the COPY BACKUP command, the START_TIME keyword works differently than it does on other RUV commands. When you specify the START_TIME keyword, the function searches the repository from that start time toward the current time for a backup to copy. Without the START_TIME keyword, the function searches the repository from the current time backward until it finds a backup to copy. The most common use of the START_TIME keyword is to cause the selection of the first backup (or set of backups, if the function is processing a group) that is later than the specified time, as opposed to the current, latest backup.

The START_TIME keyword gives you flexibility in controlling backups and DASD usage. It allows you to take several Instant Snapshot copies before copying the oldest off to tape. It allows you to obtain a coordinated set of backups for a VSAM group, even if additional backups of some data sets are taken after the set of backups that you want to coordinate.

For the START_TIME keyword value, specify the timestamp in **yyyyddhmmst** or **Xhhhhhhhhhhhhhh** format. For more information about these formats, see “Using the START_TIME Keyword” on page 111.
PREFIX Keyword

You can use the optional PREFIX keyword on the BACKUP VSAM_IN command to control whether RUV creates a prefixed backup of a VSAM KSDS. This keyword applies only to KSDS VSAM backups. For more information, see “PREFIX Keyword” on page 292.

Set one of the following values for the PREFIX keyword. The default value is YES.

YES Add the prefix to each backup registration record.

NO Do not add the prefix. This value is valid for a KSDS backup only.

OUTPUT_MODEL Keyword

For a VSAM group, use the required OUTPUT_MODEL keyword to specify the name of the output model to be used for the output copies. The output model is created with the ADD OUTPUT_MODEL command and defines how to name and allocate the output data set. You can specify multiple instances of the OUTPUT_MODEL keyword if you want multiple output copies. For more information, see “OUTPUT_MODEL Keyword” on page 297.

BACKUP_OUT Keyword

For a VSAM file (specified with the VSAM_FILE keyword), use the required BACKUP_OUT keyword to identify the ddname of the JCL statement that allocates the output copy. You can specify multiple instances of the BACKUP_OUT keyword if you want multiple output copies. For more information, see “BACKUP_OUT Keyword” on page 293.

Copy Backup Function Considerations

The following considerations apply to the Copy Backup function:

- If a VSAM group definition is changed or expanded after a backup has been created and before the Copy Backup function has been executed, the list of files that are copied might be different from the list of files that were backed up.

- If a problem occurs during execution of the Copy Backup function, no backup registration records are deleted and no original backup data sets are scratched, regardless of the values of the DELETE keyword and the SCRATCH keyword. Any completed copies are good copies. You can choose to delete the good copies manually before you rerun the Copy Backup function. No error occurs if you do not delete them, but you might have extra copies as a result.
If the Copy Backup function detects that two input data sets are on the same tape, it resets the value of the STACKED_TAPE_COUNT keyword to 1 and disables parallel processing for the remainder of the job step.

If the function detects that any data set in a VSAM group has no active VSAM backup, the function issues a warning message and return code 4 and continues to copy the remaining backups. If the function detects that no files in the group have backups to be copied, the function issues an error message and return code 8.

If the function detects that no backup exists for the specified VSAM file (as specified with the VSAM_FILE keyword), the function issues an error message and return code 8.

### Copy Backup Function Example

The following excerpt report is from a job that is used to obtain an Instant Snapshot copy of a VSAM data set. The example assumes that the ADD VSAM_FILE command and the ADD VSAM_NAME_MODEL command have been executed previously to create the required records in the repository.

```
15 BACKUP
16   VSAM_IN(RUVRUV.QA.XBM.RUVK0010)
17     VSAM_NAME_MODEL(QA_SNAP_132A):

FULL

VSAM DSN: RUVRUV.QA.XBM.RUVK0010
Backup DSN: RUVRUV.QA.XBM.RUVK0010.SNAPBKUP
Registration Date: 2009.119  20:28:31
5,000 Backup Records
```
The following excerpt report is from a job that is used to obtain a QSAM copy of the Instant Snapshot copy that was created in the previous example.

```sql
21    COPY BACKUP VSAM_FILE(RUVRUV.QA.XBM.RUVK0010)
22                INVENTORY_ONLY(NO)
23                VSAM_ONLY(YES)
24                PREFIX(YES)
25                DELETE(YES)
26                SCRATCH(YES)
27                BACKUP_IN(CURRENT_BACKUP)
28                BACKUP_OUT(BK$OUT1)
29                     COMMENT('Testing Deferred Backup')
30                LOCATION(RUV_QA)
31                STATUS(ACTIVE));
```

**Backup Copy Report**

- **VSAM DSN:** RUVRUV.QA.XBM.RUVK0010
- **From Backup DSN:** RUVRUV.QA.XBM.RUVK0010.SNAPBKUP
- **Backup DSN:** RUVRUV.QA.XBM.RUVK0010.BKUPCOPY
- **Location:** RUV_QA
- **Registration Date:** 2009.119 20:28:31
- **5,000 Backup Records**
- **489 Backup K Bytes**

Explicitly Registering Backups

The following sections provide information about explicit registration of backups in the RUV repository.

Overview of Backup Registration

RUV uses backup registration records in the repository to keep track of backups of VSAM files. In many cases, RUV performs automatic registration by creating the backup registration record for an output backup data set at the time that the backup is performed. You might want to register backups explicitly in several circumstances, including the following:

- If you use another product instead of RUV to perform a backup that is written as a VSAM file or flat file (such as produced by the IDCAMS REPRO utility), you can register this backup in the repository so that RUV can restore it automatically when necessary. Instant snapshot copies are VSAM files. In most cases, RUV registers Instant Snapshot copies automatically, but you can register them explicitly if necessary. To register a VSAM- or flat-file format backup or an Instant Snapshot copy, you use the REGISTER BACKUP_FILE command.

- If you use another product instead of RUV to perform a backup, and if you want RUV to automatically restore these backups, you can create a restore job JCL record that contains skeleton JCL for the restore process. Then you can register the backup in the repository as requiring the use of this restore job JCL record. If RUV needs to restore this type of backup, RUV automatically submits the restore job JCL record for variable substitution and execution. To register a backup as requiring the use of a restore job JCL record, you use the REGISTER RESTORE_JOB_JCL command.

- If you use another product instead of RUV to perform a backup, and if you want to manually restore this backup instead of allowing RUV to restore it automatically through a restore job JCL record, you can register the backup in the repository as requiring a restore method. When RUV needs to restore this type of backup, RUV issues a message to the operator and waits for a reply. The message specifies the name of the file to be restored, the name of the restore method, and an optional parameter that was specified when the backup was registered. To register a backup as requiring the use of a restore method, you use the REGISTER RESTORE_METHOD command.

- You can re-register an RUV backup if necessary. (The BACKUP VSAM_FILE command and the BACKUP VSAM_GROUP command can create RUV backups; this type of backup has a prefix.) For example, the backup registration record may have been removed inadvertently, or you may need to prepare a repository at an offsite location for the recovery process. To register an RUV backup, you use the REGISTER BACKUP_IN command.
Coding JCL for the REGISTER BACKUP_FILE Command

To register a non-RUV backup that has been written in VSAM or flat-file format, use the REGISTER BACKUP_FILE command as shown in the following syntax:

```
REGISTER BACKUP_FILE(ddn_or_dsn)
  VSAM_FILE(ddn_or_dsn)
  or
  SEQ_FILE(ddn_or_dsn)
  | START_TIME(yyyyjjjhhmmss | Xhhhhhhhhhhhhhhhh)
  | LOCATION(locname8) }
  | COMMENT(comment_fields) }
  | STATUS(ACTIVE | INACTIVE) }
```

**BACKUP_FILE Keyword**

Use the required BACKUP_FILE keyword on the REGISTER BACKUP_FILE command to identify the backup data set to register in the repository. You can code only one BACKUP_FILE keyword on each REGISTER BACKUP_FILE command. Set the value to the ddname of the DD statement that describes the backup data set in the JCL, or set it to the fully qualified data set name of the backup data set.

The BACKUP_FILE keyword can describe a VSAM data set, which is the type of data set that is produced as an Instant Snapshot copy. If possible, RUV works with SUF to perform an instant restore of a VSAM data set. If SUF is not available, RUV performs a VSAM-to-VSAM copy during the restore process.

**VSAM_FILE or SEQ_FILE Keyword**

Use the required VSAM_FILE or SEQ_FILE keyword on the REGISTER BACKUP_FILE command to identify the file that was backed up. You can code only one VSAM_FILE or SEQ_FILE keyword on each REGISTER BACKUP_FILE command. Set the value to the ddname of the DD statement that describes the input data set in the JCL, or set it to the fully qualified data set name of the file.

**START_TIME Keyword**

Use the optional START_TIME keyword on the REGISTER BACKUP_FILE command to override the timestamp that RUV sets (current time in GMT) for this backup in the repository. RUV can take a flat file backup and use the start time of the backup for the registration time. A non-RUV backup does not contain internal timestamp information.
If you register a backup file at a later time than when the backup was created, you should supply the time of the backup. In many cases, the current time may be current enough. For example, you back up the file in step 1 (at 3:00 AM) and register it immediately on step termination (at 3:15 AM). Either timestamp is accurate enough because no updates occurred between the two times.

Set the value to the timestamp that you want to use for this file. For more information about the values that you can code, see “Using the START_TIME Keyword” on page 111.

--- NOTE ---
RUV assumes that you are supplying all times in GMT.

--- Optional Keywords ---

You can code the following optional keywords on the REGISTER BACKUP_FILE command:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)

--- Example REGISTER BACKUP_FILE Command ---

The following example shows the registration of a non-RUV backup in the repository by using the REGISTER BACKUP_FILE command:

```
REGISTER BACKUP_FILE(SMSBKUP1)
   VSAM_FILE(GNL.LEDGER)
   START_TIME(20092582331550)
   LOCATION(LOCAL)
   COMMENT('GENERAL LEDGER SMS BACKUP')
   STATUS(ACTIVE)
```

Coding JCL for the REGISTER RESTORE_JOB_JCL Command

To register a non-RUV backup that you want RUV to restore automatically by using the JCL that you have created in a restore job JCL record, use the REGISTER RESTORE_JOB_JCL command as shown in the following syntax:

```
REGISTER RESTORE_JOB_JCL(name32)
  VSAM_FILE(ddn_or_dsn)
  START_TIME(yyyyjjjhhmmsst | Xhhhhhhhhhhhhhh)
  PARM(userfield64)
  LOCATION(locname8)
  COMMENT(comment_fields)
  STATUS(ACTIVE | INACTIVE)
```

**RESTORE_JOB_JCL Keyword**

Use the required RESTORE_JOB_JCL keyword to specify the name of the restore job JCL record that contains the JCL for restoring this backup. RUV can submit this JCL automatically during restore or recovery processing. The keyword value is the name of the restore job JCL record to use. For more information about creating a restore job JCL record, see “Working with Job JCL Records for Backup and Restore Processes” on page 327.

**VSAM_FILE Keyword**

Use the required VSAM_FILE keyword on the REGISTER RESTORE_JOB_JCL command to identify the VSAM file that was backed up. You can code only one VSAM_FILE keyword on each REGISTER RESTORE_JOB_JCL command. Set the value to the ddname of the DD statement that describes the input VSAM data set in the JCL, or set it to the fully qualified data set name of the VSAM file.

**START_TIME Keyword**

Use the optional START_TIME keyword on the REGISTER RESTORE_JOB_JCL command to override the timestamp that RUV sets (current time in GMT) for this backup in the repository. A non-RUV backup does not contain internal timestamp information.

You can use the STORE INTO_FILE command and variables to retrieve a stored time from a file. For more information, see “Storing Information into a File” on page 332.

In many cases, the current time may be current enough. For example, you back up the file in step 1 (at 3:00 AM) and register it immediately on step termination (at 3:15 AM). Either timestamp is accurate enough because no updates occurred between the two times.
Set the value to the timestamp that you want to use for this file. For more information about the values that you can code, see “Using the START_TIME Keyword” on page 111.

---

**NOTE**

RUV assumes that you are supplying all times in GMT.

---

**PARM Keyword**

Use the optional PARM keyword on the REGISTER RESTORE_JOB_JCL command to store information about the specific backup. This information can be accessed during the process of generating the restore JCL if the <PARAMETER> variable was used in the job JCL record.

The value of the PARM keyword is a user field (1 to 64 bytes).

**Optional Keywords**

You can code the following optional keywords on the REGISTER RESTORE_JOB_JCL command:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)

**Example REGISTER RESTORE_JOB_JCL Command**

The following example shows the registration of a non-RUV backup in the repository by using the REGISTER RESTORE_JOB_JCL command:

```plaintext
REGISTER RESTORE_JOB_JCL(RESTORE1)
  VSAM_FILE(GNL.LEDGER)
  START_TIME(20093200000000)
  PARM(Y)
  LOCATION(LOCAL)
  COMMENT('GENERAL LEDGER BACKUP')
  STATUS(ACTIVE)
```

Coding JCL for the REGISTER RESTORE_METHOD Command

To register a non-RUV backup that you want to restore manually at a request from RUV, use the REGISTER RESTORE_METHOD command as shown in the following syntax:

```plaintext
REGISTER RESTORE_METHOD(name32)
  VSAM_FILE(ddn_or_dsn)
   START_TIME/yyyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh
   PARM(userfield64)
   LOCATION(locname8)
   COMMENT(comment_fields)
   STATUS(ACTIVE | INACTIVE)

;  
```

**RESTORE_METHOD Keyword**

Use the required RESTORE_METHOD keyword to cause RUV to record a backup that acts as a trigger point during restore processing. When this backup is needed during restore or recovery processing, RUV issues a write-to-operator-with-reply (WTOR) message to the console to ask the operator to take action to restore the file manually. When the file is restored, the operator replies to the WTOR message to indicate the results of the manual restore process.

The value of the RESTORE_METHOD keyword is the name (1 to 32 characters) of the restore method. This name is provided in the WTOR message to help the operator understand the restore request.

The WTOR message contains the name of the restore method, the value of the PARM keyword (if any), and a list of the VSAM files to be restored with this method.

The operator can issue the following replies to the WTOR message:

- **G** (Go) All files have been successfully restored.
- **R** (Redisplay) Issue the message again to show the list of VSAM files to be restored.
- **C** (Cancel) Mark all files in the restore list as unsuccessfully restored.

**VSAM_FILE Keyword**

Use the required VSAM_FILE keyword on the REGISTER RESTORE_METHOD command to identify the VSAM file that was backed up. You can code only one VSAM_FILE keyword on each REGISTER RESTORE_METHOD command. Set the value to the ddname of the DD statement that describes the input VSAM data set in the JCL, or set it to the fully qualified data set name of the VSAM file.
START_TIME Keyword

Use the optional START_TIME keyword on the REGISTER RESTORE_METHOD command to override the timestamp that RUV sets (current time in GMT) for this backup in the repository. A non-RUV backup does not contain internal timestamp information.

You can use the STORE INTO_FILE command and variables to retrieve a stored time from a file. For more information, see “Storing Information into a File” on page 332.

In many cases, the current time may be current enough. For example, you back up the file in step 1 (at 3:00 AM) and register it immediately on step termination (at 3:15 AM). Either timestamp is accurate enough because no updates occurred between the two times.

Set the value to the timestamp that you want to use for this file. For more information about the values that you can code, see “Using the START_TIME Keyword” on page 111.

NOTE
RUV assumes that you are supplying all times in GMT.

PARM Keyword

Use the optional PARM keyword on the REGISTER RESTORE_METHOD command to store information about the specific backup. This information can be accessed during the process of generating the restore JCL if the <PARAMETER> variable was used in the job JCL record.

The value of the PARM keyword is a user field (1 to 64 bytes).

Optional Keywords

You can code the following optional keywords on the REGISTER RESTORE_METHOD command:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)
Example REGISTER RESTORE_METHOD Command

The following example shows the registration of a non-RUV backup in the repository by using the REGISTER RESTORE_METHOD command:

```
REGISTER RESTORE_METHOD(RESTORE2)
  VSAM_FILE(GNL.LEDGER)
  START_TIME(20093200000000)
  PARM(Y)
  LOCATION(LOCAL)
  COMMENT('WTOR TO RESTORE LEDGER')
  STATUS(ACTIVE)
;  
```
Coding JCL for the REGISTER BACKUP_IN Command

To register an RUV-format backup of a VSAM file (one that contains a prefix) in the repository, use the REGISTER BACKUP_IN command by using the following syntax:

```ruby
REGISTER BACKUP_IN(ddn_or_dsn)
  { LOCATION(locname8) }
  { COMMENT(comment_fields) }
  { STATUS(ACTIVE | INACTIVE) }
;
```

**BACKUP_IN Keyword**

Use the required BACKUP_IN keyword on the REGISTER BACKUP_IN command to identify the backup data set to register in the repository. You can code only one BACKUP_IN keyword on each REGISTER BACKUP_IN command. Set the value to the ddname of the DD statement that describes the backup data set in the JCL, or set it to the fully qualified data set name of the backup data set.

**Other Keywords**

You can code the following optional keywords on the REGISTER BACKUP_IN command:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)

**Example REGISTER BACKUP_IN Command**

The following example shows the registration of an RUV backup in the repository by using the REGISTER BACKUP_IN command:

```ruby
REGISTER BACKUP_IN(RUVBKUP1)
  LOCATION(LOCAL)
  COMMENT('RUV BACKUP')
  STATUS(ACTIVE) ;
```
Working with Backup Records in the Repository

The following sections provide information about updating, deleting, and reporting on backup information in the repository.

**NOTE**

RUV uses the same type of record to store backup information, whether the backup has been registered automatically by RUV or you have used the REGISTER command to store the information. You use the DELETE BACKUP_FILE command and the REPORT BACKUP_FILE command to work with all backup records.

Overview of Update, Delete, and Report Backup Functions

RUV automatically manages backup registration records in the repository. However, you might need to work with backup records as follows:

- You might need to update information about a backup data set in the repository. For example, you may need to set the primary backup to inactive and a secondary backup to active status because a hardware failure was detected on the primary backup. Or you may be preparing for an off-site recovery in which the secondary backup is the only one available, so you must set the primary backup to inactive and the secondary backup to active status.

  To update backup registration records in the repository, use the UPDATE BACKUP_FILE command as explained in “Coding JCL for the UPDATE BACKUP_FILE Command” on page 364.

- Normally, purge processing deletes backup registration records from the repository automatically. If necessary, you can delete backup registration records from the repository explicitly, for example, if the backup has become physically damaged or has been scratched.

  To delete backup registration records in the repository, use the DELETE BACKUP_FILE command as explained in “Coding JCL for the DELETE BACKUP_FILE Command” on page 367.

- RUV can provide a report about the backup registration records that are stored in the repository. You can use this type of report to determine the results of a command before the actions are performed. For example, you can run a REPORT BACKUP_FILE command to see which backup registration records would be deleted if you changed the command to DELETE BACKUP_FILE.
To obtain a report about backup registration records in the repository, use the REPORT BACKUP_FILE command as explained in “Coding JCL for the REPORT BACKUP_FILE Command” on page 369. For an example of the report that is produced by the REPORT BACKUP_FILE command, see “BACKUP_FILE Report” on page 408.

RUVP can provide a report about the backup methods that are enabled in the environment. You can use this type of report to determine whether a backup method is active and to help with diagnosis of backup problems.

For an example of the report that is produced by the REPORT BACKUP_METHOD command, see “BACKUP_METHOD Report” on page 409.

**Coding JCL for the UPDATE BACKUP_FILE Command**

You can use the UPDATE BACKUP_FILE command to update information about a backup in the repository. The command has different formats depending on whether you use the FILTER_BY keyword or the REGISTRATION_TIME keyword.

To code an UPDATE BACKUP_FILE command with the FILTER_BY keyword, use the following syntax:

```plaintext
UPDATE BACKUP_FILE(ddn_or_dsn, *, ...)  
  { FILTER_BY(  
      | LOCATION(location_name8,* , ...) )  
      | STATUS(ACTIVE | INACTIVE)  
      | START_TIME/yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh  
      | STOP_TIME/yyyyjjjhhmmssst| Xhhhhhhhhhhhhhhhh  
  )  
  { LOCATION(locname8)  
    | STATUS(ACTIVE | INACTIVE)  
    | COMMENT(comment_fields)  
  ;
```

To code an UPDATE BACKUP_FILE command with the REGISTRATION_TIME keyword, use the following syntax:

```plaintext
UPDATE BACKUP_FILE(ddn8_or_dsn44)  
  { REGISTRATION_TIME/yyyyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh  
     | LOCATION(location_name8)  
     | COMMENT('comment fields')  
     | STATUS(ACTIVE | INACTIVE)  
  ;
```
Coding JCL for the UPDATE BACKUP_FILE Command

Chapter 7 Working with Backups

**BACKUP_FILE Keyword**

Use the required BACKUP_FILE keyword on the UPDATE BACKUP_FILE command to identify the backup data set for which you want to update information in the repository. You can code only one BACKUP_FILE keyword on each UPDATE BACKUP_FILE command. Specify one of the following values:

- the ddname of the input backup data set in the JCL
- a single fully qualified data set name if you use the REGISTRATION_TIME keyword
  
  The data set name cannot contain wildcard characters, and you cannot specify multiple data set names.
- one or more fully qualified data set names, data set name masks, or both, if you do not use the REGISTRATION_TIME keyword
  
  You can use wildcard characters (\* and ?) and specify multiple values.

**FILTER_BY Keyword**

Use the optional FILTER_BY keyword to filter the selection of backup registration records to update. You can filter the selection of backup registration records by location, status, start time, and stop time. For more information, see "Using the FILTER_BY Keyword" on page 108.

If you specify the FILTER_BY keyword, do not specify the REGISTRATION_TIME keyword.

**REGISTRATION_TIME Keyword**

Use the optional REGISTRATION_TIME keyword to select a specific registered backup registration record to be modified in the repository. The value of the REGISTRATION_TIME keyword is a timestamp. You can provide the timestamp in the format \yyyy\jjj\hh\mm\ss\st or as the hexadecimal value (X\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\hh\h
Optional Keywords

You can use the following optional keywords with the BACKUP_OUT keyword on the UPDATE BACKUP_FILE command:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT “Using the COMMENT Keyword” on page 107

Example UPDATE BACKUP_FILE Commands

The following example shows the use of the UPDATE BACKUP_FILE command and FILTER_BY keyword to update information about a backup in the repository:

```
UPDATE BACKUP_FILE(LEDG*)
   FILTER_BY(
      LOCATION(LOCAL)
      STATUS(INACTIVE)
   )
   LOCATION(REMOTE)
   COMMENT('GENERAL LEDGER REMOTE BACKUP')
   STATUS(INACTIVE)
   LOCATION(REMOTE)
   COMMENT('GENERAL LEDGER REMOTE BACKUP')
   STATUS(INACTIVE)
;
```

The following example shows the use of the UPDATE BACKUP_FILE command to update a specific backup registration record based on a registration time in the repository:

```
UPDATE BACKUP_FILE(LEDGBK02)
   LOCATION(REMOTE)
   COMMENT('DISASTER RECOVERY')
   STATUS(INACTIVE)
   REGISTRATION_TIME(20092920518450)
   :
```


Coding JCL for the DELETE BACKUP_FILE Command

To delete backup registration records from the repository by using the DELETE BACKUP_FILE command, use the following syntax:

```
DELETE BACKUP_FILE(ddn_or_dsn_or_'internal name', *, ..)
{ FILTER_BY(
    [ LOCATION(location_name8,*,...) ]
    [ STATUS(ACTIVE | INACTIVE) ]
    [ START_TIME(yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh) ]
    [ STOP_TIME(yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh) ]
)
{ SCRATCH(NO | YES) }
```

**BACKUP_FILE Keyword**

Use the required BACKUP_FILE keyword on the DELETE BACKUP_FILE command to identify the backup registration records to delete. The value can be the ddname of the backup file in the JCL, the data set name of the backup file, or the RUV-generated internal name of the backup record (if the backup was registered with the REGISTER RESTORE_JOB_JCL command or the REGISTER RESTORE_METHOD command). If you specify a data set name or the RUV-generated name, you can use masking (wildcard) characters. You can code multiple values for this keyword.

The DELETE BACKUP_FILE command does not read the backup data set. It uses the name you supply only to identify the record to delete in the repository. However, if you specify the ddname, z/OS allocation of the data set occurs. It is recommended that you use the data set name instead of the ddname so that z/OS allocation does not occur.

RUV generates the following internal name for a backup record that you have created with the REGISTER RESTORE_JOB_JCL command (--JOB_JCL is followed by three blanks):

```
--JOB_JCL your_job_jcl_name
```

RUV generates the following internal name for a backup record that you have created with the REGISTER RESTORE_METHOD command (--WTOR is followed by six blanks):

```
--WTOR your_restore_method_name
```
If you specify the specific name (instead of a mask) in the DELETE command, you must enclose the complete name in single quote marks because the name contains embedded blanks. See the following examples:

```sql
DELETE BACKUP_FILE('--JOB_JCL EXT002B_RESTORE');
DELETE BACKUP_FILE('--WTOR EXT002B_WTO');
```

**FILTER_BY Keyword**

Use the optional FILTER_BY keyword to filter the selection of backup registration records to be deleted. You can filter the selection of backup registration records by location, status, start time, and stop time. For more information, see “Using the FILTER_BY Keyword” on page 108.

**SCRATCH Keyword**

Use the optional SCRATCH keyword on the DELETE BACKUP_FILE command to control whether to remove the backup data set from the z/OS catalog. The default value is NO.

- **NO**  Do not remove the backup file from the z/OS catalog.
- **YES**  Remove the backup file from the z/OS catalog.

**NOTE**

If RUV does not find and delete a repository record (it receives return code 4), it takes no further action to scratch the data set even if you specify SCRATCH(YES) and the z/OS data set exists.

**Example DELETE BACKUP_FILE Command**

The following example shows the deletion of backup records from the repository:

```sql
DELETE BACKUP_FILE(GNL.LEDGBK*, GNL.PAYRBK*)
    FILTER_BY(
        START_TIME(20092460000000)
    )
    SCRATCH(YES)
;```
To report information about the backup registration records in the repository by using the REPORT BACKUP_FILE statement, use the following syntax:

```
REPORT BACKUP_FILE(ddn_or_dsn_or_'internal name', *, ..)
{ FILTER_BY(
    { LOCATION(location_name8,*, ...) }
    { STATUS(ACTIVE | INACTIVE) }
    { START_TIME(yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhh) }
    { STOP_TIME(yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhh) }
    )
]}
```

### BACKUP_FILE Keyword

Use the required BACKUP_FILE keyword on the REPORT BACKUP_FILE command to identify the backup registration records to be reported. The value can be the ddname of the backup file in the JCL, the data set name of the backup file, or the RUV-generated internal name of the backup record (if the backup was registered with the REGISTER RESTORE_JOB_JCL command or the REGISTER RESTORE_METHOD command). If you specify a data set name or the RUV-generated name, you can use masking (wildcard) characters. You can code multiple values for this keyword.

The REPORT BACKUP_FILE command does not read the backup data set. It uses the name you supply only to identify the backup registration record in the repository. However, if you specify the ddname, z/OS allocation of the data set occurs. It is recommended that you use the data set name instead of the ddname so that z/OS allocation does not occur.

RUV generates the following internal name for a backup record that you have created with the REGISTER RESTORE_JOB_JCL command (--JOB_JCL is followed by three blanks):

```
--JOB_JCL your_job_jcl_name
```

RUV generates the following internal name for a backup record that you have created with the REGISTER RESTORE_METHOD command (--WTOR is followed by six blanks):

```
--WTOR your_restore_method_name
```
If you specify the specific name (instead of a mask) in the REPORT command, you must enclose the complete name in single quote marks because the name contains embedded blanks. See the following examples:

```
REPORT BACKUP_FILE('--JOB_JCL   EXT002B_RESTORE');
REPORT BACKUP_FILE('--WTOR   EXT002B_WTO');
```

**Using the FILTER_BY Keyword**

Use the optional FILTER_BY keyword to filter the selection of backup registration records to be reported. You can filter the selection of backup registration records by location, status, start time, and stop time. For more information, see “Using the FILTER_BY Keyword” on page 108.

**Example REPORT BACKUP_FILE Command**

The following example shows the use of the REPORT BACKUP_FILE command to obtain a report about backup files:

```
REPORT BACKUP_FILE(GNL.LEDGBK*, GNL.PAYRBK*)
  FILTER_BY(
    START_TIME(20092460000000)
  )
;  
```
Coding JCL for the REPORT BACKUP_METHOD Command

To report information about the backup methods that are enabled in the environment by using the REPORT BACKUP_METHOD command, use the following syntax:

```
REPORT BACKUP_METHOD( * | RUV BWO SMS SNAPSHOT EXTERNAL) ;
```

**BACKUP_METHOD Keyword**

Use the required BACKUP_METHOD keyword on the REPORT BACKUP_METHOD command to identify the backup methods to be reported. The following values are valid. The default value is * (asterisk). If you do not code *, you can code multiple values for this keyword.

* Provide information about all backup methods in the environment.

**RUV**

Provide information about the RUV backup method.

**BWO**

Provide information about the Backup-While-Open backup method.

**SMS**

Provide information about the SMS backup method.

**SNAPSHOT**

Provide information about the Instant Snapshot copy backup method.

**EXTERNAL**

Provide information about the external backup method.

**Example REPORT BACKUP_METHOD Command**

The following example shows the use of the REPORT BACKUP_METHOD command to report information about all backup methods that are enabled in the environment:

```
REPORT BACKUP_METHOD(*) ;
```
Creating Reports

This chapter introduces the RUV reports. The following topics are included:

**Reporting RUV Data with ISPF Panels**
- VSAM DSN List .......................... 376
- Applid/Fileid List ......................... 378
- VSAM DSN Detail - Base Cluster Panel .... 379
- DSN Detail - Applid/Fileid List Panel ... 380
- DSN Detail - User Recovery Exit List Panel 381
- DSN Detail - Comments Pop-up .......... 382
- DSN Detail - Path List Panel ............ 383
- DSN Detail - Backup List Panel .......... 384
- Delete/Define Statements in ISPF Edit Panel 385
- Archive DSN List ......................... 387
- Backup File List ......................... 389
- Log of Logs List ......................... 391
- Groups / Output / VSAM Models ......... 393
- Subsystem / Ruleset Information ....... 395
- Message Lookup ......................... 398
- RUV Job Monitoring ..................... 400

**Reporting RUV Data in Batch**
- Selecting a Batch Report Level .......... 402
- Obtaining RUV Batch Reports .......... 403
- ARCHIVE_FILE Report ................. 405
- APPLID Report ......................... 407
- BACKUP_FILE Report ................... 408
- BACKUP_METHOD Report ............... 409
- DEFAULT Report ......................... 410
- EXTERNAL_VENDOR Report .............. 411
- INTERNAL_READER Report ............ 412
- JOB_JCL Report ........................ 413
- JOB_SET Report ......................... 414
- JOURNAL_MODEL Report ............... 415
- LOG_OF_LOGS Report ................... 416
- LOG_STREAM Report ...................... 418
- OUTPUT_MODEL Report .................. 419
- RULE_SET Report ....................... 420
<table>
<thead>
<tr>
<th>Report Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM_VARIABLES Report</td>
<td>422</td>
</tr>
<tr>
<td>UNIT_OF_WORK Report</td>
<td>423</td>
</tr>
<tr>
<td>USER_VARIABLES Report</td>
<td>424</td>
</tr>
<tr>
<td>VSAM_FILE Report</td>
<td>425</td>
</tr>
<tr>
<td>VSAM_GROUP Report</td>
<td>427</td>
</tr>
<tr>
<td>VSAM_GROUP_DEFINITION Report</td>
<td>428</td>
</tr>
<tr>
<td>VSAM_NAME_MODEL Report</td>
<td>429</td>
</tr>
<tr>
<td>VSAM_SET Report</td>
<td>430</td>
</tr>
<tr>
<td>XBMID Report</td>
<td>431</td>
</tr>
<tr>
<td>RUV REPORT AUDIT</td>
<td>432</td>
</tr>
<tr>
<td>Using the RUV REPORT AUDIT Function</td>
<td>433</td>
</tr>
<tr>
<td>Using the REPORT AUDIT Command</td>
<td>434</td>
</tr>
<tr>
<td>Selecting VSAM Files</td>
<td>434</td>
</tr>
<tr>
<td>Using Optional Keywords</td>
<td>435</td>
</tr>
<tr>
<td>INVENTORY_ONLY Keyword</td>
<td>435</td>
</tr>
<tr>
<td>START_TIME Keyword</td>
<td>435</td>
</tr>
<tr>
<td>STOP_TIME Keyword</td>
<td>435</td>
</tr>
<tr>
<td>TERMID Keyword</td>
<td>436</td>
</tr>
<tr>
<td>TRANID Keyword</td>
<td>436</td>
</tr>
</tbody>
</table>
Reporting RUV Data with ISPF Panels

RUV can display ISPF panels or create batch reports that show repository contents. Repository contents include VSAM data sets selected for recovery, APPLID and FILEID data, archived data sets, backup-file data sets, log of log data, and rule set information. ISPF panels provide repository information that is available during a TSO session.

To access the RUV Options panel (Figure 19 on page 115), start the ISPF interface with procedures described in “Starting the RUV ISPF Interface” on page 113. After the RUV Options panel is displayed, you can obtain reports by selecting options 2 through 8.
VSAM DSN List

To display the VSAM DSN List panel, select option 2 on the RUV Options panel. The VSAM DSN List panel (Figure 72) displays VSAM DSN names in ascending order.

The VSAM DSN List panel shows you VSAM data sets that are collected and registered to RUV. RUV will recover these VSAM data sets. Data sources for this list may include CICS logs and journals, Transaction Server log streams and the log of logs, RUV batch journaling and backup, or the manual registration of a VSAM file.

Figure 72 VSAM DSN List Panel

To display information that is related to recovery resources associated with a specific DSN, select the DSN and press Enter.

- If you selected a base cluster, the VSAM DSN Detail - Base Cluster panel is displayed. For more information, see “VSAM DSN Detail - Base Cluster Panel” on page 379.

- If you selected a path, the DSN Detail - Path List panel is displayed. For more information, see “DSN Detail - Path List Panel” on page 383.
You can display a subset of the VSAM files by applying a filter to the VSAM DSN List panel. Select View and then Filter on the action bar to display the pop-up panel (Figure 73). The filters may include text or wildcards.

**Figure 73 Filter Pop-Up Panel**
Applid/Fileid List

To display the Applid/Fileid List panel, select option 3 on the RUV Options panel. The Applid/Fileid List panel (Figure 74) displays APPLID and FILEID names in ascending order.

The Applid/Fileid List panel displays the APPLIDs and FILEIDs that are associated with every VSAM base cluster or path that is updated by CICS or Transaction Server. Data sources are CICS logs and journals, Transaction Server log streams, the log of logs, and manual input.

In a batch process you can enter APPLID and FILEID associations manually when you are adding or updating a VSAM DSN.

**Figure 74   Applid/Fileid List Panel**

<table>
<thead>
<tr>
<th>Appliance</th>
<th>Fileid</th>
<th>VSAM DSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPCEBASE</td>
<td>RUVRCM/QA0.RPCEBASE</td>
<td></td>
</tr>
<tr>
<td>RPCEPATH</td>
<td>RUVRCM/QA0.RPCEPATH</td>
<td></td>
</tr>
<tr>
<td>RPCEO010</td>
<td>RUVRCM/QA0.RPCEO010</td>
<td></td>
</tr>
<tr>
<td>RPCKBASE</td>
<td>RUVRCM/QA0.RPCKBASE</td>
<td></td>
</tr>
<tr>
<td>RPCKDPV0</td>
<td>RUVRCM/QA0.RPCKDPV0</td>
<td></td>
</tr>
<tr>
<td>RPCKPATH</td>
<td>RUVRCM/QA0.RPCKPATH</td>
<td></td>
</tr>
<tr>
<td>RPCRKP4</td>
<td>RUVRCM/QA0.RPCRKP4</td>
<td></td>
</tr>
<tr>
<td>RPCK0010</td>
<td>RUVRCM/QA0.RPCK0010</td>
<td></td>
</tr>
<tr>
<td>RPCK0020</td>
<td>RUVRCM/QA0.RPCK0020</td>
<td></td>
</tr>
<tr>
<td>RPCR0010</td>
<td>RUVRCM/QA0.RPCR0010</td>
<td></td>
</tr>
<tr>
<td>RUVEBASE</td>
<td>RUVRUV/DV711VP.RUVEBASE</td>
<td></td>
</tr>
<tr>
<td>RUVEBASE</td>
<td>RUVRUV/QA.IAM.RUVEBASE</td>
<td></td>
</tr>
</tbody>
</table>

To display additional detail about the associated VSAM file, select an item and press Enter. The next panel that is displayed depends on the type of VSAM file that you select:

- If you select a base cluster file, the VSAM DSN Detail - Base Cluster panel is displayed (Figure 75).
- If you select a path, the DSN Detail - Path List panel is displayed (Figure 79).
VSAM DSN Detail - Base Cluster Panel

When you select a base cluster file on the VSAM DSN List panel or the Applid/Fileid List panel, the VSAM DSN Detail - Base Cluster panel is displayed (Figure 75). Use this panel to view basic information about the VSAM file.

Figure 75 VSAM DSN Detail - Base Cluster Panel

You can select one or more choices in the lower half of the panel to obtain more information about the file, as explained in the following sections.
When you select **Applid/Fileid list** on the VSAM DSN Detail - Base Cluster panel, the DSN Detail - Applid/Fileid List panel is displayed (Figure 76). This choice displays APPLID and FILEID pairs that are associated with the base cluster data set name.

**Figure 76   DSN Detail - Applid/Fileid List Panel**
DSN Detail - User Recovery Exit List Panel

When you select User exit list on the VSAM DSN Detail - Base Cluster panel, the DSN Detail - User Recovery Exit List panel is displayed (Figure 77). This choice displays information about the user exit that is associated with the base cluster data set name.

Figure 77  DSN Detail - User Recovery Exit List Panel
When you select Comments on the VSAM DSN Detail - Base Cluster panel, the VSAM DSN Detail - Comments pop-up is displayed (Figure 78). This choice displays the comments that were registered along with the base cluster data set name.

Figure 78  DSN Detail - Comments Panel
DSN Detail - Path List Panel

When you select Path list on the VSAM DSN Detail - Base Cluster panel, or when you select a path on the Applid/Fileid List panel, the DSN Detail - Path list panel is displayed (Figure 79). Use this panel to view information about the paths that are associated with the base cluster.

Figure 79 DSN Detail - Path List Panel

On this panel, VSAM key attributes and the upgrade set status are displayed for paths that are associated with the base VSAM cluster. RUV can obtain information from any of the following sources:

Catalog
- the VSAM catalog

Sphere
- sphere information that was saved when the file was backed up

Rpstry
- data contained in the repository record for the path DSN itself.
When you select **Backup list** on the VSAM DSN Detail - Base Cluster panel, the DSN Detail - Backup List panel is displayed (Figure 80). This choice displays the list of backups for the selected VSAM data set. The backup data set names are displayed in registration-time order with the most current backup data set at the top of the list.

**Figure 80  DSN Detail - Backup List Panel**
Delete/Define Statements in ISPF Edit Panel

When you select Delete/Define on the VSAM DSN Detail - Base Cluster panel (Figure 75 on page 379), an ISPF Edit panel (Figure 81) displays the IDCAMS DELETE/DEFINE statements for the data set.

**Figure 81  Delete/Define Statements in ISPF Edit Panel**

RUV can obtain the information to create the DELETE/DEFINE statements from the following sources:

- the system catalog
- the RUV repository

If the information is available in both the system catalog and in the repository, RUV displays a pop-up window for you to select the source that you want to use. If the information is not available from either source, RUV displays a message saying that no DELETE/DEFINE information is available.

You can edit any of the displayed statements and save the changes. In this way, you can use the selected VSAM file as a model for creating or changing other files.
You can enter the PERDD command to execute the DELETE/DEFINE commands. A message is displayed to indicate whether the DELETE/DEFINE operation has been successful.

**WARNING**

When you enter the PERDD command, the file that is named in the statements is deleted and defined immediately. Ensure that you have specified the correct file name to prevent the inadvertent deletion of the wrong file.
Archive DSN List

To display the Archive DSN List panel, select option 4 on the RUV Options menu. The Archive DSN List (Figure 82) displays the names of archive DSN files containing information that is used to perform a recovery or accumulation.

When you process the ARCHIVE command against a CICS journal or a Transaction Server log stream, an RUV archive file is created and the DSN name will appear in this list. An archive file is created by RUV for each batch step that is being journaled. Archives are registered and displayed in on this panel using the Registration Time of the most current record in the file.

Figure 82 Archive DSN List Panel

The Status of the archives is displayed.

- ACTIVE archives can be used for a recovery or an accumulation.
- INACTIVE archives are not selected by RUV for use in a recovery.
You can display information about a specific archive by selecting the DSN name and pressing Enter. For example, **Figure 83** is the Archive Detail pop-up panel showing that there are 570 records that can be recovered in the ACTIVE archive.

**Figure 83  Archive DSN Detail Panel**

Based on your permission level, you can select more detailed information about the archive. In this example, by selecting the options in the Archive Detail pop-up, you can display the VOLSER List information or the Job Detail Information as shown in **Figure 84**.

**Figure 84  Additional DSN Archive Data**
Backup File List

To display the Backup DSN List panel, select option 5 on the RUV Options menu. The Backup DSN List panel (Figure 85) displays the names of VSAM files that were created and registered when running an RUV BACKUP command.

The Backup DSN List panel shows DSNs that were created and registered when the RUV BACKUP command is run. RUV uses these registered backups during automated restore operations for VSAM files. Backups are registered and displayed on this panel using the starting time of the most current backup as the Registration Time.

Figure 85  Backup DSN List Panel

The Status of the archives is displayed.

- ACTIVE (active) archives can be used for a recovery.
- INACT (inactive) archives are not selected by RUV for use in a recovery.

You can display information about a specific Backup DSN by selecting the DSN name and pressing Enter. For example, Figure 86 is the Backup Detail panel showing information on the VSAM base cluster and the VSAM DSN file that was backed up.
Based on your permission level, you can select more detailed information about the VSAM data sets. In this example, by selecting the Job Information option, you can display the Backup Detail - Job Information panel as shown in Figure 87.

**Figure 86  Backup Detail Panel**

![Backup Detail Panel](image)

**Figure 87  Backup Detail - Job Information Panel**

![Backup Detail - Job Information Panel](image)
Log of Logs List

To display the Log of Logs Selection panel, select option 6 on the RUV Options menu. The Log of Logs Selection panel (Figure 88) displays log of logs stream names in alphabetical sequence.

The Log of Logs Selection panel shows the name and status of z/OS log streams are to be scanned automatically when a log stream is archived. Automatic scanning of a log of logs stream updates the APPLID and FILEID to VSAM DSN associations for Transaction Server systems.

**Figure 88  Log of Logs Selection Panel**

The Status of all the log streams is displayed and reflects how each should be handled by RUV.

- **Active (active)** — Selecting the log stream begins the immediate scanning and registration of log stream data.

- **Inact (inactive)** — The log of logs record is not registered to z/OS as a log stream and is not automatically scanned or updated. To use an inactive log stream, it must be updated to active and registered as a log of logs in a batch process.
To display information from the log stream for a specific log of logs, select the log stream name and press Enter. The Log of Logs Detail panel is displayed (Figure 89).

**Figure 89  Log of Logs Detail Display Panel**

The Transaction Server records the open and close of each VSAM file in the log of logs and displays the information in ascending order based on the VSAM DSN.
Groups / Output / VSAM Models

To display the Groups / Output / VSAM Model panel, select option 7 on the RUV Options menu. The Groups / Output / VSAM Model panel (Figure 90) displays the list of group and output model names.

Select the View > Sort option on the action bar to sort the list on Group–Model Name or Type.

Figure 90 Group and Output Model List Panel
To display information about the models, select the individual name and press Enter. Typical details for each type are shown in the following figures.

**Figure 91  Details for Output Model**

```
OUTPUT_MODEL(GRP_MODEL_19_2)
  Comment("This is the output_model for GRPO019"
           "This model is used for the second copy")
  BACKUP_OUT_TYPE(DASD)
  QSAM("DSN=<VSAM_DSN>.<RUVBACKUP>.
        "SPACE=(CYL,(5,1),RLSE)."
        "UNIT=SYSALLDA,DISP=(NEW,CATLG)."
    )
```

**Figure 92  Details for VSAM Group**

```
VSAM_GROUP_DEFINITION(BWO_GROUP)
  VSAM_FILE(RUVRCM.SP.REC.RCMK0010,INCLUDE)
  VSAM_FILE(ROHMXN.LESSOR.L4AD,INCLUDE)
  VSAM_FILE(RUVRV.QA.DFHCMACD,INCLUDE)
```

**Figure 93  Details for VSAM_NAME_MODEL**

```
VSAM_Name_model(SSI_F00)
  Comment(" BACKUP METHOD MODEL SNAPOUT1"
    "REPLACE( YES)"
  )
  VSAM_File_mask (RUVRUV.RUVIN.*
                  RUVRUV.RUVOUT.<NODE3>.<NODE4>
  )
```

---

**.Recovery Utility for VSAM User Guide**
To display the Subsystem Detail panel, select option 8 on the RUV Options menu.

The Subsystem Detail panel (Figure 94) displays information on all active subsystem levels including: the subsystem version; the subsystem start time; the operating system version; the subsystem ID (SSID) of the subsystem currently connected; the system ID (SMFID) of the system where the subsystem is running, and the current backup and logging states in the connected subsystem.

**Figure 94  Subsystem Detail Panel**
On the Subsystem Detail panel, select the Rule Set option and press Enter to display the current Rule Set List Panel (Figure 95).

**Figure 95  Rule Set List Panel**

![Rule Set List Panel](image)

For information on working with rule sets, see “Using the ISPF Interface to Work with Rule Sets” on page 205.

To view the rule set, on the Rule Set List Panel, select the specific rule set and press Enter. The contents of the rule set is displayed in the ISPF TSO editor (Figure 96).
Figure 96  Sample of Rule Set in List Panel

```plaintext
Menu  Utilities  Compiliers  Help

BROWSE    RDHMJS.SI.RUV  Line 00000000 Col 001 080
********************************************************** Top of Data **********************************************************
RULE_SET($DEFAULT)
   JOB_SET($DEFAULT,
      $EXCLUDE_ALL)
   VSAM_Set($DEFAULT)
   User_Variables($DEFAULT)
   Comment("BMC RUV Default Rule Set *)
;
********************************************************** Bottom of Data **********************************************************

Command ===>
```
Message Lookup

To display the RUV Message Lookup panel, select option 9 on the RUV Options menu. The Message Lookup panel (Figure 97) enables you to browse all messages or select views indexed by internal message number, external message number, or message module.

Figure 97  Message Lookup Panel

Choose the type of search, and type a search value.

- 1. Message number. Wildcard characters cannot be used in the search value.
- 2. Text. Wildcard characters (?, *, or both) can be used in the search value.
- 3. Internal Number. This option is intended for use at the direction of BMC Software.

To initiate a message search, press Enter. The Message Text panel (Figure 98) is displayed.
The information about the message, includes the module that issues the message, the offset within the module, the internal message number (for BMC Software use), the message text, a description of the message, and the action that you should take to address the message.
RUV Job Monitoring

To display the RUV Job Monitoring panel, select option 11 on the RUV Options panel. The RUV Job Monitoring panel (Figure 99) displays information about the RUV jobs that are currently executing on the system image. This information relates to the regions that are associated with RUV activities, including batch jobs being journaled by RUV, Backup-While-Open regions associated with RUV, and RUV utility jobs.

This information can help you determine which jobs would be affected if you bring down the subsystem.

The same information is displayed on the lower half of the Subsystem Detail panel (Figure 94 on page 395).

Figure 99  RUV Job Monitoring Panel
Reporting RUV Data in Batch

RUV produces a variety of batch reports that show repository contents. These contents include VSAM data sets selected for recovery, APPLID and FILEID data, archived data sets, backup file data sets, log of log data, system variables and user variables, and rule set information.

You should use reports to display various types of data created in your environment including the current state of the data. First use the following RUVZSM0 batch interface JCL. Then use the REPORT command to display repository data (for further JCL setup explanation, see Chapter 3, “Using RUV Interfaces.”)

```plaintext
//RUVZSM0 EXEC PGM=RUVZSM0
//*STEPLIB DD ==>IF REQUIRED
//SYSPRINT DD SYSOUT=* 
//SYSIN DD * 
```

Most RUV reports accept lists of values as parameters. These values may contain wildcard characters. Parameters allow you to restrict the data to the information you need to see. Use the following command to set a wildcard search for all instances of a specified report type:

```plaintext
REPORT report_type(*);
```

Many of the RUV reports also accept filter keywords. Filter keywords allow you to further restrict the selection of data that you are interested in. The keywords START_TIME, STOP_TIME, and LOCATION are accepted as filters for reporting archive and backup file data as shown in the following example:

```plaintext
SET REPORT(1evel);
  REPORT report_type(*)
    START_TIME(start_time)
    STOP_TIME(stop_time)
    LOCATION(location);
```
The REPORT command can be especially useful before you delete information from the repository (recommended). Use the REPORT command as previously shown to examine data before deletion from the repository. When you are satisfied with what will be deleted, change the REPORT keyword to the DELETE keyword. Rerun the job to delete the previously reported information as shown in the following example:

```plaintext
SET REPORT(level);
   DELETE report_type(*)
   FILTER_BY(
       START_TIME(start_time)
       STOP_TIME(stop_time)
       LOCATION(location));
```

### Selecting a Batch Report Level

You can select report levels that produce summary, detail, or full reports. For some reports, the contents of reports change depending on the reporting level that you specify. For other reports, the contents remain the same for any level requested. If you do not specify a level, the default is a summary report. Batch reports usually sort key data in alphabetical order. Use the SET command to specify reporting level as shown in the following example:

```plaintext
SET REPORT(SUMMARY|DETAIL|FULL);
```

### Summary Reports

If no level is specified, summary is the default reporting level. Summary reporting typically provides you with a high-level view of repository information. The amount of information varies depending on the type of data being reported. For example, a summary of the REPORT APPLID command lists all CICS or Transaction Server APPLIDs in the RUV repository as shown in the following example:

```plaintext
REPORT APPLID(*);
Applid: QACICS42
Applid: QACICS43
```
Detail Reports

Detailed reports provide expanded repository field information. This expansion may be the preferred level for reports that you work with on a day-to-day basis. As shown in the following example, if you run the REPORT APPLID command with the SET REPORT(DETAIL) command, the data is expanded to show you all VSAM DSNs associated with the APPLID and FILEID:

| APPLID: QACICS42 FILEID: OADCLGA VSAM DSN: RUV.OADCLGA |
| APPLID: QACICS42 FILEID: OADCTF VSAM DSN: RUV.OADCTF  |
| APPLID: QACICS42 FILEID: OADLST VSAM DSN: RUV.OADLST  |
| APPLID: QACICS42 FILEID: OADNEG VSAM DSN: RUV.OADNEG  |
| APPLID: QACICS42 FILEID: OADQUE VSAM DSN: RUV.OADQUE  |
| APPLID: QACICS43 FILEID: CPADC VSAM DSN: RUV.CPADC  |
| APPLID: QACICS43 FILEID: CPCHKM VSAM DSN: RUV.CPCHKM  |
| APPLID: QACICS43 FILEID: CPCHKS VSAM DSN: RUV.CPCHKS  |
| APPLID: QACICS43 FILEID: CPPAX VSAM DSN: RUV.CPAX   |
| APPLID: QACICS43 FILEID: DDXVACF VSAM DSN: RUV.VACF   |

Full Reports

Full reports typically provide a printout of all fields for the selected repository records. Select this level with discretion because full reports tend to be large.

Obtaining RUV Batch Reports

Perform the following steps for running an RUV batch report:

1. Optional. Select which RUV subsystem you wish to report on. (If you have only a single subsystem, the subsystem will be selected automatically by RUV.)

2. Determine the area of interest that needs to be displayed and select the desired reporting level (summary, detail, full). Summary is the default level.

Table 15 shows the RUV reports available for printing and the reporting level available for each.
3 Add appropriate control statements to the batch JCL procedures described in Chapter 3, “Using RUV Interfaces.”

4 Run the job and review the output.
ARCHIVE_FILE Report

The ARCHIVE_FILE report contains record-level recovery information you will need to perform recoveries. Archive files are created each time you run ARCHIVE commands against a CICS journal, Transaction Server log stream, or run RUV batch journaling. Archives are registered by the time of the most current record on the file. (Inactive archives are not selected automatically for use in recovery.) The ARCHIVE_FILE report in Figure 100 shows the archives in ascending time sequence.

Figure 100  ARCHIVE_FILE Summary Report Sample

```plaintext
2009.015 20:09:03 GMT   BMC RECOVERY UTILITY for VSAM -- Version 04.01.00                  Page: 1
2009.015 15:09:03 Local    COPYRIGHT BMC SOFTWARE, INC. 2009                       Report Level: Summary

1  SET SUBSYSTEM(VQA4);            
2  SET REPORT(SUMMARY);           
3  REPORT ARCHIVE_FILE(*)         
4  FILTER_BY(                     
5      START_TIME(2009005000000)  
6      STOP_TIME(2009005010000));  

Registration Date: 2009.005  00:02:15  
Status: Active  
Job: RUVI731A  JES: J0B06255  Userid: RCHFES2  
Stepname: COPYIVP1  Pgm: IDCAMS    XCF: ESAJ      SMFID: ESAJ

Archive: RUVRUV.DV731VP.RUVI731A.JOURNAL1.JOURNAL  
Registration Date: 2009.005  00:05:05  
Status: Active  
Job: RUVI731A  JES: J0B06255  Userid: RCHFES2  
Stepname: JOURNAL1  Pgm: VSPGM     XCF: ESAJ      SMFID: ESAJ

Registration Date: 2009.245  00:02:15  
Status: Active  
Job: RUVI731A  JES: J0B06255  Userid: RCHFES2  
Stepname: COPYIVP1  Pgm: IDCAMS    XCF: ESAJ      SMFID: ESAJ
```

Figure 101 provides a detail report sample of archive files in the repository.

Figure 101  ARCHIVE_FILE Detail Report Sample

```plaintext
2009.015 20:09:03 GMT   BMC RECOVERY UTILITY for VSAM -- Version 04.01.00                  Page: 1
2009.015 15:09:03 Local    COPYRIGHT BMC SOFTWARE, INC. 2009                       Report Level: Detail

1  SET SUBSYSTEM(VQA4);            
2  SET REPORT(DETAIL);             
3  REPORT ARCHIVE_FILE(*)          
4  FILTER_BY(                      
5      START_TIME(2009005000000)  
6      STOP_TIME(2009005010000));  

Registration Date: 2009.245  00:02:15  
Status: Active  
Job: RUVI731A  JES: J0B06255  Userid: RCHFES2  
Stepname: COPYIVP1  Pgm: IDCAMS    XCF: ESAJ      SMFID: ESAJ
```

Archive characteristics
Device Name: 3390
SMS Managed: No
Gap Detected: No
Date range: 2009.245 00:00:46 to: 2009.245 00:02:15
Figure 102 provides a full report sample of archive files in the repository.

Figure 102  ARCHIVE_FILE Full Report Sample

```
2009.015 20:09:03 GMT     BMC RECOVERY UTILITY for VSAM -- Version 04.01.00       Page: 1

1  SET SUBSYSTEM(VQA4);
2  SET REPORT(FULL);
3  REPORT ARCHIVE_FILE(*);
4    FILTER_BY(
5      START_TIME(2009005000000)
6      STOP_TIME(2009005010000));

Archive: RUVRUV.RUVI731A.02009245.T0000463.J
Registration Date: 2009.245  00:02:15
Status: Active
Job: RUVI731A  JES: J0B06255  Userid: RCHFES2
Stepname: COPYIVP1  Pgm: IDCAMS  XCF: ESAJ  SMFID: ESAJ
Archive characteristics
Device Name: 3390
SMS Managed: No
Gap Detected: No
Date range: 2009.245 00:00:46 to: 2009.245 00:02:15
200 Log records
200 Journal records

VSAM DSNs on this archive
DSN: RUVRUV.DV73IVP1.RUVK0010
Status: Active
Applid: RUVI731A Fileid: F0000041
Date range: 2009.245 00:00:46 to: 2009.245 00:00:50
20 Log records
20 Journal records
```
The APPLID report shows the association of APPLIDs and FILEIDs with every VSAM base cluster or path that is updated by CICS logs and journals, Transaction Server log streams, and the log of logs. The APPLID report in Figure 103 shows the APPLIDs and FILEIDs in alphabetical order.

**Figure 103  APPLID Detail Report Sample**

```
1  SET SUBSYSTEM(VQA4);
2  SET REPORT(DETAIL);
3  REPORT APPLID(*):

Applid: RUVQA11B   Fileid: RUVKPTH2   VSAM DSN: RUVRUV.QA.BAK.RUVKPTH2
Applid: RUVQA11B   Fileid: RUVKRKP9   VSAM DSN: RUVRUV.QA.BAK.RUVKRKP9
Applid: RUVQA11B   Fileid: RUVKSM50  VSAM DSN: RUVRUV.QA.BAK.RUVKSM50
Applid: RUVQA11B   Fileid: RUVK0010  VSAM DSN: RUVRUV.QA.BAK.RUVK0010
Applid: RUVQA11B   Fileid: RUVRSMS0  VSAM DSN: RUVRUV.QA.BAK.RUVRSMS0
Applid: RUVQA11B   Fileid: RUVRV010   VSAM DSN: RUVRUV.QA.BAK.RUVRV010
```
The BACKUP_FILE report shows DSNs created during an RUV BACKUP command. RUV (which registers backups when a BACKUP command is processed) uses registered backups during automated restore operations for VSAM files. The registration date and time is the starting time of the backup. Inactive backups are not selected automatically for use in restore or recovery. The BACKUP_FILE report in Figure 104 shows the backup files in ascending time sequence.

For more information about backup names that can be reported, see “Coding JCL for the REPORT BACKUP_FILE Command” on page 369.

Figure 104  BACKUP_FILE Summary Report Sample

```
2009.015 20:09:03 GMT  BMC RECOVERY UTILITY for VSAM -- Version 04.01.00                  Page: 1
2009.015 15:09:03 Local  COPYRIGHT BMC SOFTWARE, INC. 2009                       Report Level: Summary

1   SET SUBSYSTEM(VQA4);
2   SET REPORT(SUMMARY);
3    REPORT BACKUP_FILE(*)
4    FILTER_BY(
5      START_TIME(2009005000000)
6      STOP_TIME(2009005010000));

Backup: RUVRUV.DV741VP.RUVEBASE.BKUP
VSAM DSN: RUVRUV.DV741VP.RUVEBASE
Status: Active
Type: Flat
Registration Date: 2009.245  00:41:50
Volume serial number(s):  DEV585
Job: RUVI741C  JES: JOB06267  Userid: RCHFES2
Stepname: BKUP     Procstep: RUVBATCH  Pgm: RUVZSM0   XCF: ESAJ    SMFID:ESAJ

Backup: RUVRUV.DV741VP.RUVESMS0.BKUP
VSAM DSN: RUVRUV.DV741VP.RUVESMS0
Status: Active
Type: Flat
Registration Date: 2009.245  00:40:46
Volume serial number(s):  DEV515
Job: RUVI741C  JES: JOB06267  Userid: RCHFES2
Stepname: BKUP     Procstep: RUVBATCH  Pgm: RUVZSM0   XCF: ESAJ    SMFID:ESA
```
The BACKUP_METHOD report provides information about the backup methods that are enabled in the environment. The BACKUP_METHOD report in Figure 105 shows the type of information that is produced when you issue the REPORT BACKUP_METHOD(*) command.

Figure 105 BACKUP_METHOD Report Sample

```
2009.348 13:46:24 GMT        BMC RECOVERY UTILITY for VSAM -- Version 04.01.00          Page: 1
1   SET SUBSYSTEM(VDEV);
2   SET REPORT(DETAIL);
3   REPORT BACKUP_METHOD(*);
RUV V02.01.00 active
BWO Information
   No BWO participants
   BWO Coordinator: BCSS using SVC 255
SMS Information
   HSM: active
   SMS: 1.5.0
Snapshot Information
   No Snapshot systems are defined to RUV
DEFAULT is EXTERNAL_BACKUP
RUV202140I RUV EOJ. Maximum CC was 0 .
```
DEFAULT Report

The DEFAULT report shows your user-specified default settings in the repository. Predefined defaults are supplied with the base RUV product. You create your own defaults by subsystem with the SET DEFAULT command. These defaults are stored in the repository. If you do not set these defaults, you will receive a “RUV202094I RUV No default options are set” message in the report. You restore the base defaults with the RESET DEFAULT command. (For information about default-related commands, see Chapter 3, “Using RUV Interfaces.”) The DEFAULT report in Figure 106 shows user-specified defaults.

Figure 106 DEFAULT Summary Report Sample

```
2009.015 20:09:03 GMT   BMC RECOVERY UTILITY for VSAM -- Version 04.01.00                  Page: 1
2009.015 15:09:03 Local   COPYRIGHT BMC SOFTWARE, INC. 2009                       Report Level: Summary

1  SET SUBSYSTEM(VQA4);
2  REPORT DEFAULT;

LINE_COUNT ( 78 )
DATE_FORMAT ( DD/MM/YYYY )
```
EXTERNAL_VENDOR Report

The EXTERNAL_VENDOR report shows the user exit program name. (For information on external vendors, see Chapter 4, “Using the Batch Journaling Facility.”) The EXTERNAL_VENDOR report in Figure 107 shows external vendor names in alphabetic order.

Figure 107 EXTERNAL_VENDOR Summary Report Sample

<table>
<thead>
<tr>
<th>2009.015 16:05:19 GMT</th>
<th>BMC RECOVERY UTILITY for VSAM – Version 04.01.00</th>
<th>Page: 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 REPORT EXTERNAL_VENDOR(*)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTERNAL VENDOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMENT ('50 CHARACTERS OF COMMENTS SINGLE QUOTE DELINIATED'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>' FOR UP TO 4 LINES'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>' AND FOLLOWED BY'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>' A TERMINATING PARENTHESIS'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BACKUP_REGISTRATION (NO)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOURNAL_REGISTRATION (ABD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'64 CHARACTER PARAMETER STRING'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATUS (ACTIVE) ;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
INTERNAL_READER Report

The INTERNAL_READER report shows the name of the internal readers (rule set elements) you defined that are available to use for job step backout processing. The internal reader defines the partitioned data set (PDS) name and member name to store backout JCL; or, it identifies a JES job class for immediate submission of the JCL. (For information about internal readers, see Chapter 4, “Using the Batch Journaling Facility.”) The INTERNAL_READER report in Figure 108 shows the internal reader names in alphabetical order.

Figure 108  INTERNAL_READER Summary Report Sample

```
1   SET SUBSYSTEM(VQA4);
2   SET REPORT(SUMMARY);
3    REPORT INTERNAL_READER(*);

Internal Reader: $DEFAULT
Class (A)

Internal Reader: IVP_INTRDR_JES
Class (A)

Internal Reader: IVP_INTRDR_PDS
   PDS_Out (<TESTCASE_DSN>)
      Member (<TEST_TYPE>RECOV)

Internal Reader: IVP731_INTRDR_JES
Class (A)

Internal Reader: IVP731_INTRDR_PDS
   PDS_Out (<TESTCASE_DSN>)
      Member (<TEST_TYPE>RECOV)
```
JOB_JCL Report

The JOB_JCL report shows the name of the skeleton JCL (a rule set element) that you may use for building backout JCL and control statements. (For information about skeleton JCL elements, see Chapter 4, “Using the Batch Journaling Facility.”) The JOB_JCL report in Figure 109 shows the job JCL names in alphabetical order.

Figure 109  JOB_JCL Summary Report Sample

```
2009.015 20:09:03 GMT       BMC RECOVERY UTILITY for VSAM -- Version 04.01.00     Page: 1
2009.015 15:09:03 Local        COPYRIGHT BMC SOFTWARE, INC. 2009                  Report Level: Summary

1   * Environmental set up
2   SET REPORT(SUMMARY);
3   REPORT JOB_JCL(*);

Job JCL: $DEFAULT
   //<JOBNAME>  JOB (<ACCOUNT>),JOURNAL RECOVERY',CLASS=A,
   //     NOTIFY=&SYSUID,MSGCLASS=X,REGION=4M
   //*
   //RECOVER EXEC PGM=RUVZSM0
   //DFSPARM DD *
   LIST,LISTX,MSGPRT=ALL,STZE=MAX-40K
   //SORTPARM DD *
   MSG=AP,CORE=MAX-40K, DYNALLOC=ON
   //SYSPRINT DD SYSOUT=*  
   //SYSIN DD *
   <RUV_CONTROL>

Job JCL: COMPANY TEST JCL
   //<USERID(1,6)>RJ JOB (4540),MSGCLASS=X,CLASS-A,
   //     REGION=4M, NOTIFY=&SYSUID
   /* THIS WOULD BE RECOVERY JOB FOR <JOBNAME>
   // SYSCONF = &SYSCLONE
   // SYSTYPE = &SYSNAME
   // SYSPLEX = &SYSPLEX
   // DATE: <RUVDATE(2,7)> TIME: <RUVTIME(2,7)>  
   // <RUV LOG>
   /* <RUV_JOURNAL>
   //RUVRECOV EXEC PGM=RUVZSM0
   //SYSIN DD *
   <RUV_CONTROL>
```
JOB_SET Report

The JOB_SET report shows the job sets that you will use in a job rule. (For information about job sets in rule set construction, see Chapter 4, “Using the Batch Journaling Facility.”) The JOB_SET report in Figure 110 shows the JOB_SET names in alphabetical order.

Figure 110  JOB_SET Summary Report Sample

| 1   | SET SUBSYSTEM(VQA4);          |
| 2   | SET REPORT(SUMMARY);          |
| 3   | REPORT JOB_SET(*);            |

Job set: $DEFAULT
Comment ( Standard set of RUV log/journal excludes
BMC Software
)
Job: *
   Program: DFH*
   Log (EXCLUDE)
   Journal (EXCLUDE)
   Journal_ABEND (ABEND)
   Job_ABEND (SUBMIT)
   Job_JCL ($DEFAULT)
   Internal_reader ($DEFAULT)
Job: *
   Program: DFS*
   Log (EXCLUDE)
   Journal (EXCLUDE)
   Journal_ABEND (ABEND)
   Job_ABEND (SUBMIT)
   Job_JCL ($DEFAULT)
   Internal_reader ($DEFAULT)
The JOURNAL_MODEL report shows the journal models you will use to create templates for constructing a DD allocation statement for QSAM logs and journals. (For information about journal models, see Chapter 4, “Using the Batch Journaling Facility.”) The JOURNAL_MODEL report in Figure 111 shows the journal model names in alphabetical order.

Figure 111  JOURNAL_MODEL Summary Report Sample

```
1   SET SUBSYSTEM(VQA4);
2   SET REPORT(SUMMARY);
3   REPORT JOURNAL_MODEL(*);

Journal Model: QATEST_JOURNAL
   Comment ( Internal BMC Testing                      
               BMC Software )                          
   QSAM (DSN=RUVRUV.<JOBNAME>.<RUVDATE>.<RUVTIME>.J,  
          SPACE=(CYL,(<PRIMARY_SPACE>,<SECONDARY_SPACE>),RLSE),  
          DISP=(MOD,DELETE,CATLG),  
          UNIT=<DASD>)                 
   PROTECTION_LEVEL(1)

Journal Model: QATEST_LOG
   Comment ( Internal BMC Testing                      
               BMC Software )                          
   QSAM (DSN=RUVRUV.<JOBNAME>.<RUVDATE>.<RUVTIME>.L,  
          SPACE=(CYL,(<PRIMARY_SPACE>,<SECONDARY_SPACE>),RLSE),  
          DISP=(MOD,DELETE,CATLG),  
          UNIT=<DASD>)                 
   PROTECTION_LEVEL(1)
```
The LOG_OF_LOGS report displays information on z/OS log streams that are automatically scanned when a log stream is archived. (For information on the LOG_OF_LOGS keyword, see “Managing the Transaction Server Log of Logs” on page 239) The LOG_OF_LOGS report, set to SUMMARY, in Figure 112 shows the journal model names and status.

**Figure 112 SUMMARY LOG_OF_LOGS Report Sample**

<table>
<thead>
<tr>
<th>2009.015 20:09:03 GMT</th>
<th>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00                  Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.015 15:09:03 Local</td>
<td>COPYRIGHT BMC SOFTWARE, INC. 2009</td>
</tr>
<tr>
<td></td>
<td>Report Level: Summary</td>
</tr>
</tbody>
</table>

1  SET SUBSYSTEM(VQA4);
2  SET REPORT(SUMMARY);
3  REPORT LOG_OF_LOGS(*);

Log of Logs: RUV.DV.CICS$ALL.LOGOFLOG
Status: Active

Log of Logs: RUVQAALL.CICS$ALL.LOGOFLOG
Status: Active
The LOG_OF_LOGS report, set to FULL, in Figure 113 shows more detail about the same data sets.

### Figure 113  FULL LOG_OF_LOGS Report Sample

<table>
<thead>
<tr>
<th>Report Date</th>
<th>Time</th>
<th>GMT/Local</th>
<th>Report Level</th>
<th>Status</th>
<th>Last Scanned</th>
<th>Start Block</th>
<th>Date Range</th>
<th>Job</th>
<th>JES</th>
<th>Userid</th>
<th>Job Date</th>
<th>Stepname</th>
<th>Procstep</th>
<th>Pgm</th>
<th>XCF</th>
<th>SMFID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.321 21:20:59</td>
<td>15:20:59 Local</td>
<td>04.01.00</td>
<td>Summary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 SET SUBSYSTEM(BCSS);</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 SET REPORT(FULL);</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 REPORT LOG_OF_LOGS(*);</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Log of Logs: RCM.QA.CICSXXX.LOGOFLOG
- **Status:** Active
- **Last scanned:** 2006.065 17:05:59 GMT, 2006.065 11:05:59 Local
- **Start Block:** 0000.000 00:00:00 GMT, 0000.000 00:00:00 Local
- **Date range:** 2006.065 17:22:12 - 2006.065 17:22:12 GMT
- **Job:** RUVBATCH Proctstep: BATCH Pgm: RUVZSM0 XCF: ESAJ SMFID: ESAJ

#### Log of Logs: THIS_IS_A_TEST
- **Status:** Active
- **Comment:** JUST A TEST

#### Log of Logs: RUV.DV.CICS$ALL.LOGOFLOG
- **Status:** Active
- **Last scanned:** 2006.086 15:47:31 GMT, 2006.086 09:47:31 Local
- **Start Block:** 0000.000 00:00:00 GMT, 0000.000 00:00:00 Local
- **Date range:** 2006.085 17:42:12 - 2006.085 17:42:12 GMT
- **Job:** RUV$ARC Proctstep: TS$ARCH Pgm: RUVZSM0 XCF: SYSO SMFID: SYSO

#### Log of Logs: RUVQAALL.CICS$ALL.LOGOFLOG
- **Status:** Active
- **Last scanned:** 2006.086 15:47:33 GMT, 2006.086 09:47:33 Local
- **Start Block:** 0000.000 00:00:00 GMT, 0000.000 00:00:00 Local
- **Date range:** 2006.086 15:34:14 - 2006.086 15:34:14 GMT
  - 2006.086 09:34:14 - 2006.086 09:34:14 Local
- **Job:** RUV$ARC Proctstep: TS$ARCH Pgm: RUVZSM0 XCF: SYSO SMFID: SYSO

RUVO2021401 RUV EOJ. Maximum CC was 0.
The Log Stream report (Figure 114) displays the data set name and status of each log stream that you select with the REPORT LOG_STREAM command. You can use this information to determine whether a particular log stream is registered in the repository and whether it is active or inactive.

Figure 114  LOG_STREAM Summary Report Sample

<table>
<thead>
<tr>
<th>2009.015 20:09:03 GMT</th>
<th>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00</th>
<th>Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.015 15:09:03 Local</td>
<td>COPYRIGHT BMC SOFTWARE, INC. 2009</td>
<td>Report Level: Summary</td>
</tr>
</tbody>
</table>

13  ADD LOG_STREAM(RUV.LOGSTREAM1);
14  REPORT LOG_STREAM(*);

Logstream Name: RUV.LOGSTREAM1
Status: Active

Logstream Name: RUV.LOGSTREAM2
Status: Active
The OUTPUT_MODEL report (Figure 115) displays the output model name and associated allocation information which is used when this model name is requested. You can use this information to determine the definition assigned to a model name.

Figure 115  OUTPUT_MODEL Summary Report Sample

```
1   SET SUBSYSTEM(VGRP);
2   SET REPORT(SUMMARY);
3   REPORT OUTPUT_MODEL(*);

Output Model: GRP_MODEL_19
  Comment ( This is the output_model for GRP0019
                This one is used for the first copy )
  QSAM (DSN=<VSAM_DSN>.B,
       SPACE=(CYL,(5,1),RLSE),
       UNIT=SYSALLDA,DISP=(NEW,CATLG),
       VOL=SER=DEV9XX)
  BACKUP_OUT_TYPE(DASD)

Output Model: GRP_MODEL_19_2
  Comment ( This is the output_model for GRP0019
                This model is used for the second copy )
  QSAM (DSN=<VSAM_DSN>,<RUVBACKUP>,
       SPACE=(CYL,(5,1),RLSE),
       UNIT=SYSALLDA,DISP=(NEW,CATLG),
       VOL=SER=DEV9XX)
  BACKUP_OUT_TYPE(DASD)
  STACKED_TAPE_COUNT(5)

Output Model: GRP_MODEL_19_3
  Comment ( This is the output_model for GRP0019
                This model is used for the third copy )
  QSAM (DSN=<VSAM_DSN>..<+1>,
       SPACE=(CYL,(5,1),RLSE),
       UNIT=SYSALLDA,DISP=(NEW,CATLG),
       VOL=SER=DEV9XX)
  BACKUP_OUT_TYPE(DASD)
  STACKED_TAPE_COUNT(5)

Output Model: GRP_MODEL_19_4
  Comment ( This is the output_model for GRP0019
                This model is used for the fourth copy )
  QSAM (DSN=RUVRUV..<VSAM_DSN(-15)>..<+1>.B,
       SPACE=(CYL,(5,1),RLSE),
       UNIT=SYSALLDA,DISP=(NEW,CATLG),
       VOL=SER=DEV9XX)
  BACKUP_OUT_TYPE(DASD)
  STACKED_TAPE_COUNT(1)
```
The RULE_SET report shows the VSAM sets, job sets, journal models, recovery JCL, and user variables that you combined into rule sets to use in various RUV batch processes. (For information about rule sets, see Chapter 4, “Using the Batch Journaling Facility.”) The RULE_SET report in Figure 116 shows the rule set names in alphabetical order.

Figure 116 RULE_SET Summary Report Sample

| 1   | * Environmental set up |
| 2   | SET REPORT(SUMMARY); |
| 3   | REPORT RULE_SET(*); |

Rule set: $DEFAULT
  Comment ( BMC RUV Default Rule Set )
  Job_set ($DEFAULT $EXCLUDE_ALL)
  VSAM_set ($DEFAULT)
  User_variables ($DEFAULT)

Rule set: $EXCLUDE_ALL
  Job_set ($EXCLUDE_ALL)
  VSAM_set ($EXCLUDE_ALL)
  User_variables ($DEFAULT)

Rule set: JNL003_RULES
  Job_set (JNL_TEST_BOTH)
  VSAM_set (JNL_TEST_BOTH)
  User_variables (JNL_VARS)

Rule set: V11TEST
  Comment ( BMC RUV Default Rule Set )
  Job_set (V11TEST $EXCLUDE_ALL)
  VSAM_set (V11TEST $EXCLUDE_ALL)
  User_variables (V11TEST $DEFAULT)
Figure 117 provides a detail report sample of rule set listings in the repository.

### Figure 117  RULE_SET Detail Report Sample

<table>
<thead>
<tr>
<th>2009.015 20:09:03 GMT</th>
<th>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00</th>
<th>Page: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.015 15:09:03 Local</td>
<td>COPYRIGHT BMC SOFTWARE, INC. 2009</td>
<td>Report Level: Detail</td>
</tr>
</tbody>
</table>

```
1  * Environmental set up
2  SET REPORT(DETAIL):
3  REPORT RULE_SET(*);

VSAM_SET ($DEFAULT)
USER_VARIABLES ($DEFAULT)

JOB SET: $DEFAULT
  COMMENT ( STANDARD SET OF RUV JOB/PROGRAM EXCLUDES
            BMC Software         )
  JOB: *          PROGRAM: DFH*
    LOG (EXCLUDE)
    JOURNAL (EXCLUDE)
    JOURNAL_ABEND (ABEND)
    JOB_ABEND (SUBMIT)
    JOB_JCL ($DEFAULT)
    INTERNAL_READER ($DEFAULT)
  JOB: *          PROGRAM: IDCAMS
    LOG (EXCLUDE)
    JOURNAL (EXCLUDE)
    JOURNAL_ABEND (ABEND)
    JOB_ABEND (SUBMIT)
    JOB_JCL ($DEFAULT)
    INTERNAL_READER ($DEFAULT)
  JOB: *          PROGRAM: DFS*
    LOG (EXCLUDE)
    JOURNAL (EXCLUDE)
    JOURNAL_ABEND (ABEND)
    JOB_ABEND (SUBMIT)
    JOB_JCL ($DEFAULT)
    INTERNAL_READER ($DEFAULT)
  JOB: *          PROGRAM: DXP*
    LOG (EXCLUDE)
    JOURNAL (EXCLUDE)
    JOURNAL_ABEND (ABEND)
```

Chapter 8  Creating Reports 421
The SYSTEM_VARIABLE report shows all preset RUV system settings. (For information on system variables, see Chapter 4, “Using the Batch Journaling Facility.”) The SYSTEM_VARIABLE report in Figure 118 shows all preset RUV system variables.

**Figure 118  SYSTEM_VARIABLES Summary Report Sample**

```
1 SET SUBSYSTEM(VQA4);
2 SET REPORT(SUMMARY);
3 REPORT SYSTEM_VARIABLES;
System Variables
<RUVDATE>/D2009245/
<RUVTIME>/T2016074/
<SMFID>/SYSP/
<JOBNAME>/HLK$RPT/
<PROCNAME>/REP$1/
<STEPNAME>/REP$1/
<PGMNAME>/RUVZSM0/
<USERID>/HLK/
<JOB_NUMBER>/JOB02687/
<RUV_JOURNAL>/NULLFILE/
<RUV_LOG>/NULLFILE/
<RUV_CONTROL>/ /
<+1>/V0000217/
RUV2021401 RUV EOJ. Maximum CC was 0 .
```
UNIT_OF_WORK Report

The UNIT_OF_WORK report shows information about the completed and incomplete transactions in the UOW archive. The UNIT_OF_WORK report in Figure 119 shows all complete and incomplete transactions in the UOW archive.

Figure 119  UNIT_OF_WORK Report Sample

<table>
<thead>
<tr>
<th>Unit_Of_Work: ROHMXN.ARCHAD7.DFHLOGA.COPYTS13.INFL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registration Date: 2009.269 16:51:15 GMT</td>
</tr>
<tr>
<td>2009.269 11:51:15 Local</td>
</tr>
<tr>
<td>Status: Active</td>
</tr>
<tr>
<td>Comment ( TEST - COPY ts1/ls UOW jnl )</td>
</tr>
<tr>
<td>Stepname: TEST  Procstep: TEST  Pgm: RUVZSM0  XCF: ESAJ  SMFID: ESAJ</td>
</tr>
<tr>
<td>Archive characteristics</td>
</tr>
<tr>
<td>Device Name: 3390</td>
</tr>
<tr>
<td>SMS Managed: No</td>
</tr>
<tr>
<td>11 Complete unit of work records</td>
</tr>
<tr>
<td>8 Incomplete unit of work records</td>
</tr>
<tr>
<td>11 total Complete unit of work records</td>
</tr>
<tr>
<td>8 total Incomplete unit of work records</td>
</tr>
<tr>
<td>VSAM DSNs on this archive</td>
</tr>
<tr>
<td>RUV202140I RUV EOJ. Maximum CC was 0.</td>
</tr>
</tbody>
</table>
The USER_VARIABLES report shows the RUV variable names and the variables defined with the ADD USER_VARIABLES command. (For information on user variables, see Chapter 4, “Using the Batch Journaling Facility.”) The USER_VARIABLES report in Figure 120 shows user-specified variables.

Figure 120  USER_VARIABLES Summary Report Sample

```
1  SET SUBSYSTEM(VQA4);
2  SET REPORT(SUMMARY);
3  REPORT USER_VARIABLES(*);

User Variables: $DEFAULT
  Comment ( Sample RUV user variables
          Use with RUV $DEFAULT rule sets
          BMC Software )
  <DASD>            /WORK/
  <PRIMARY_SPACE>   /15/
  <SECONDARY_SPACE> '5'
  <ACCOUNT>         '4540'

User Variables: IVP_VARS
  Comment ( BMC IVP VARIABLES )
  <YEAR>    /<RUVDATE(2,4)>/
  <JULDAY>  /<RUVDATE(6,3)>/
  <HOUR>    /<RUVTIME(2,2)>/
  <MINUTE>  /<RUVTIME(4,2)>/
  <SECOND>  /<RUVTIME(6,2)>/
  <PRIMARY> /2/
  <SECONDARY> /3/
  <ACCOUNT> /4540/
  <TEST_TYPE> /IVP/
  <TESTCASE_DSN> /RUVRUV.QA.V11.JCL/
```
The VSAM_FILE report shows registered VSAM DSN records with associated repository information. (For information about the VSAM_FILE keyword, see Chapter 3, “Using RUV Interfaces.”) The VSAM_FILE report in Figure 121 shows a summary of registered VSAM DSNs in alphabetical order.

**Figure 121  VSAM_File Summary Report Sample**

|-----------------------|---------------------------------------------------|------------------------|---------------------------------|-----------------------|

1   * Environmental set up
2   SET REPORT(SUMMARY);
3   REPORT VSAM_file(*):

VSAM file: RUVRUV.QA.API.RUVEBASE
File Type: ESDS
Format: Fixed
DAYS(30)

VSAM file: RUVRUV.QA.API.RUVESMSO
File Type: ESDS
Format: Variable
Using subsystem default purge value

VSAM file: RUVRUV.QA.API.RUVE0010
File Type: ESDS
Format: Fixed
CYCLES(4)
The VSAM_FILE report in Figure 122 shows more detail of registered VSAM DSNs. Detail level reports provide an inventory of associated recovery assets (backups and archives).

Figure 122  VSAM_File Detail Report Sample
The VSAM_GROUP report shows registered VSAM Group records with associated repository information. (For information about the VSAM_GROUP keyword, see Chapter 15, “Using VSAM Groups.”) The VSAM_GROUP report in Figure 123 shows a detail of a registered VSAM Group.

Figure 123  VSAM_GROUP Detail Report Sample

```plaintext
1  * Environmental set up
2  SET REPORT(DETAIL);
3  REPORT VSAM_GROUP(*);

VSAM file: RUVRUV.QA_GRP.RUVEBASE
File Type: ESDS
Format: Fixed
Using subsystem default purge value
Path Name: RUVRUV.QA_GRP.RUVEPTH1
Dataset has been changed since its last backup.
Backup DSN: RUVRUV.QA_GRP.RUVEBASE.V0000067
  Location: On_site2
  Status: Inactive
  Type: Fuzzy
  Registration Date: 2009.263  23:32:45
Volume serial number(s):  DEV387
Backup DSN: RUVRUV.QA_GRP.RUVEBASE.D2009263.T2312AA
  Location: Off_site
  Status: Inactive
  Type: Fuzzy
  Registration Date: 2009.263  23:32:45
Volume serial number(s):  DEV301
Backup DSN: RUVRUV.QA_GRP.RUVEBASE.B
  Status: Active
  Type: Fuzzy
  Registration Date: 2009.263  23:32:45
Volume serial number(s):  DEV320
```
The VSAM_GROUP_DEFINITION report shows registered VSAM Group definitions with associated repository information. (For information about the VSAM_GROUP keyword, see Chapter 15, “Using VSAM Groups.”) The VSAM_GROUP report in Figure 124 shows a detail of a registered VSAM Group.

Figure 124  VSAM_GROUP_DEFINITION Detail Report Sample

```
1   * Environmental set up
2   SET REPORT(DETAIL);
3   REPORT VSAM_GROUP_DEFINITION(*);

Comment ( This is the group definition for GRP0019 )
RUVRUV.QA.GRP.RUV* INCLUDE
RUVRUV.QA.GRP.RUVKMAXI EXCLUDE

VSAM_GROUP_DEFINITION FOR GRP_TEST_20
Comment ( This is the group definition for GRP0020 )
RUVRUV.QA.GRP.RUV* INCLUDE
RUVRUV.QA.GRP.RUVKMAXI EXCLUDE

VSAM_GROUP_DEFINITION FOR GRP_TEST_21
Comment ( This is the group definition for GRP0021 )
RUVRUV.QA.GRP.RUV* INCLUDE
RUVRUV.QA.GRP.RUVKMAXI EXCLUDE

VSAM_GROUP_DEFINITION FOR GRP_TEST_22
Comment ( This is the group definition for GRP0022 )
RUVRUV.QA.GRP.RUV* INCLUDE
RUVRUV.QA.GRP.RUVKMAXI EXCLUDE

VSAM_GROUP_DEFINITION FOR GRP_TEST_23
Comment ( This is the group definition for GRP0023 )
RUVRUV.QA.GRP.RUV* INCLUDE
RUVRUV.QA1.GRP.RUV* INCLUDE
RUVRUV.QA2.GRP.RUV* INCLUDE
RUVRUV.QA3.GRP.RUV* INCLUDE
RUVRUV.QA4.GRP.RUV* INCLUDE
RUVRUV.QA5.GRP.RUV* INCLUDE
RUVRUV.QA.GRP.RUVKMAXI EXCLUDE

VSAM_GROUP_DEFINITION FOR GRP_TEST_24
Comment ( This is the group definition for GRP0024 )
RUVRUV.QA.GRP.RUV* INCLUDE
RUVRUV.QA.GRP.RUVKMAXI EXCLUDE
```
The VSAM_NAME_MODEL report (Figure 125) displays the definitions that have been coded for VSAM name models. For information about VSAM name models, see “VSAM_NAME_MODEL Keyword” on page 305.

Figure 125  VSAM_NAME_MODEL Detail Report Sample

```
2009.255 21:35:49 GMT  BMC RECOVERY UTILITY for VSAM -- Version 04.01.00  Page: 1

1  SET SUBSYSTEM(VFS1);
2  SET REPORT(DETAIL);
3  REPORT VSAM_NAME_MODEL(*);

VSAM_Name_model: FES_TEST_GEN1
  Replace (NO)
  VSAM_File_mask ( +
                      RCHFES3.<NODE1>.<NODE2>.<NODE3>.<+1> )

VSAM_Name_model: IVP741D_VNM1
  Comment ( TESTDSN=<VSAM_DSN> )
  Replace (NO)
  VSAM_File_mask ( +
                      RUVRUV.DV74ISP.*
                      <NODE1>.<NODE2>.<NODE3>.BKUP1 )
```
The VSAM_SET report shows the VSAM_SET names in the repository. Each set contains VSAM journaling rules that determine which journal records, if any, to write to the journal on an individual DSN basis. (For information on VSAM sets, see Chapter 4, “Using the Batch Journaling Facility.”) The VSAM_SET report in Figure 126 shows a summary of VSAM_SET names and the VSAM rules associated with each name in alphabetical order.

Figure 126  VSAM_SET Summary Report Sample

```
2009.015 20:09:03 GMT  BMC RECOVERY UTILITY for VSAM -- Version 04.01.00  Page: 1
2009.015 15:09:03 Local  COPYRIGHT BMC SOFTWARE, INC. 2009  Report Level: Summary

1   SET SUBSYSTEM(VQA4):
2   SET REPORT(SUMMARY):
3   REPORT VSAM_SET(*);

VSAM set: $DEFAULT
   Comment ( RUV default - exclude everything )
   VSAM_rule (* Exclude)

VSAM set: $EXCLUDE_ALL
   Comment ( Exclude all files BMC Software )
   VSAM_rule (* Exclude)

VSAM set: QATEST
   Comment ( Internal BMC Testing BMC Software )
   VSAM_rule (RVARAD* Exclude)
   VSAM_rule (RUVASF* Exclude)
   VSAM_rule (RUVBSL* Exclude)
   VSAM_rule (*DUPLEX* Exclude)
```
The XBMID report (Figure 127) displays information about the EXTENDED BUFFER MANAGER (XBM) subsystems that SNAPSHOT UPGRADE FEATURE (SUF) uses when processing Instant Snapshot copies. If a particular XBM subsystem is active, the report shows the version, release, and maintenance level of the XBM subsystem.

Figure 127   XBMID Report Sample

1  SET SUBSYSTEM(VFS1);
2  SET REPORT(DETAIL);
3  REPORT XBMID(XVRA XVRC XVRG XVRF XVR1);

XBMID: XVRA   VERSION: 5.3.00
XBMID: XVRC   VERSION: 5.3.00
XBMID: XVRF   VERSION: 5.3.00
XBMID: XVRG   VERSION: 5.3.00
XBMID: XVR1   VERSION: 5.3.00

RUV202141I RUV Maximum CC set to 4.
RUV202140I RUV EOJ. Maximum CC was 4.
Auditing has grown in importance over the last few years, both for reporting to the
government (as in the Sarbanes-Oxley requirements) or simply for better
management.

The RUV REPORT AUDIT feature is designed to help in that process by allowing
reports to be generated that cover changes to VSAM data sets by processing Archives
alone. The feature is designed to be easy to use by anyone with access to the RUV
system, not only by programmers.

As long as the Archives are still registered in the Repository, the RUV REPORT
AUDIT selection process can be completely automatic. At its simplest, you only need
to tell the job the VSAM data set name and the time on which to report, and it will do
the rest. It will select the correct Archives and generate the report. If the Archive was
from CICS, it will report on the terminal ID and transaction ID, along with all the
other usual information (key, time of change, type of change, etc.). It is not necessary
to access the actual data set (or for it to even exist at report time) to get a report, as
long as the Archives are available.

For more specialized reports, the data can be filtered by terminal ID and/or
transaction ID, as well as VSAM data set and time of change. For example, it is a
simple process to get a report on the changes made to a VSAM data set (or group of
VSAM data sets) made from one particular terminal ID during a specific 22-minute
period on the 12th of July last year (as long as the Archives are still available).
Using the RUV REPORT AUDIT Function

To use the RUV REPORT AUDIT function, perform the following steps:

1. Determine the data on which to REPORT AUDIT.

   The REPORT AUDIT function can select individual VSAM files, or it can process all of the files in a VSAM group that you have defined with the ADD or UPDATE VSAM_GROUP_DEFINITION command (as described in “Defining Groups” on page 601).

   It can also be given an Archive as input and told to report on all the files in that Archive.

2. Code the JCL to execute the REPORT AUDIT function of the RUVZSM0 utility.

   - Request REPORT AUDIT for individual VSAM files with the REPORT AUDIT VSAM_FILE command.
   - Request REPORT AUDIT for a defined VSAM group with the REPORT AUDIT VSAM_GROUP command.
   - Request REPORT AUDIT for a specific Archive with the REPORT AUDIT ARCHIVE_IN command.

3. Submit the JCL for execution.

4. Check the results of the execution.
Using the REPORT AUDIT Command

Report on the changes to a VSAM file by using the REPORT AUDIT command, as shown in Figure 128:

**Figure 128  REPORT AUDIT Command Example**

```
REPORT AUDIT
  VSAM_FILE(GNL.LEDGER)
  VSAM_GROUP(GLGROUP)
  ARCHIVE_IN(GNL.LEDGER.ARCH01)
  INVENTORY_ONLY(NO)
  START_TIME(20082890000001)
  STOP_TIME(20082900000001)
  TERMID(0020)
  TRANID(QA01)

;  
```

**Selecting VSAM Files**

REPORT AUDIT reports changes made to the file that is specified by the VSAM_FILE parameter. Archives are automatically selected and processed for the report. No data sets are changed by this process, and only the Archives are read (not the VSAM_FILE).

REPORT AUDIT reports changes made to the files that are specified by the VSAM_GROUP parameter. The data set names are identified in the VSAM group definition that must have been previously defined. Archives are automatically selected and processed for the report. No data sets are changed by this process, and only the Archives are read (not the VSAM files).

REPORT AUDIT reports changes made to the files in the Archives that are specified by the ARCHIVE_IN parameter. Only Archives that are specified are selected and processed for the report. No data sets are changed by this process, and only the Archives are read (not the VSAM files).
Using Optional Keywords

You can further refine the changes reported on by adding the optional keywords. Keywords on the REPORT AUDIT command filter the file to make the report more specific. The following sections describe the valid keywords in the order that they are listed.

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the REPORT AUDIT VSAM_FILE command to produce a report of the resources required without actual execution of the Report process. Because no execution occurs, no data sets are accessed if you use this option.

Set one of the following values for the INVENTORY_ONLY keyword.

The default value is NO.

**NO**  Produce an Audit Report without a preliminary resource report.

**YES**  Produce a preliminary report of VSAM and Archive file resources, but do not perform the Report process.

START_TIME Keyword

Use the optional START_TIME keyword on the REPORT AUDIT command to specify the start (earliest) timestamp of the range of Archive records to include in the Report process.

For a REPORT AUDIT, this timestamp is optional. Use it as needed to filter the changes to be reported. The default is to use all records.

For more information, see “Using the START_TIME Keyword” on page 111.

STOP_TIME Keyword

You can use the optional STOP_TIME keyword on the REPORT AUDIT command to specify the stop (latest) timestamp of the range of Archive records to include in the Report process.

For more information, see “Using the STOP_TIME Keyword” on page 110.
**TERMID Keyword**

You can use the optional TERMID keyword on the REPORT AUDIT command to specify the ID of the terminals to use in the range of Archive records to include in the report process. Terminals are only recorded if the Archive is from CICS.

**TRANID Keyword**

You can use the optional TRANID keyword on the REPORT AUDIT command to specify the ID of the CICS transaction to use in the range of Archive records to include in the report process. Transactions are only recorded if the Archive is from CICS.
Reorganizing VSAM Files

This chapter describes how to use RUV for reorganizing VSAM files.

The following information is included:

Overview ......................................................... 438
Using the RUV REORGANIZE Function ......................... 439
  Reorganizing Individual VSAM Files .......................... 440
    Input VSAM Files ........................................... 440
    REORGANIZE VSAM_IN Command ............................. 440
      VSAM_IN Keyword ......................................... 440
      INVENTORY_ONLY Keyword ................................ 441
      BACKUP_OUT Keyword ..................................... 441
      Other Optional Keywords .................................. 441
      AIX Keyword .............................................. 442
      RETAIN_BACKUP Keyword .................................. 442
      VSAM_NAME_MODEL Keyword ................................. 442
  Reorganizing VSAM Groups .................................... 443
    Input VSAM Files ........................................... 443
    Output Backup Data Sets ................................... 444
    REORGANIZE VSAM_GROUP Command ............................ 444
      VSAM_GROUP Keyword ...................................... 444
      INVENTORY_ONLY Keyword .................................. 444
      OUTPUT_MODEL Keyword .................................... 445
      Other Optional Keywords .................................. 445
      AIX Keyword .............................................. 445
      RETAIN_BACKUP Keyword .................................. 445
      VSAM_NAME_MODEL Keyword .................................. 446
Overview

VSAM files need to be reorganized regularly to clean up CA/CI splits that can hurt performance. The RUV REORGANIZE command automates that process. The REORGANIZE command first does a BACKUP, followed immediately by a RESTORE. The RESTORE deletes the data set, and then sequentially restores it with no splits. You can decide to keep the backup data set for future use, or automatically delete it after the RESTORE is complete (see the “RETAIN_BACKUP Keyword” on page 442).

If you have SNAPSHOT installed and specify the VSAM_NAME_MODEL (SNAPSHOT) parameter, RUV REORGANIZE uses SNAPSHOT for the BACKUP phase, but does not use SNAPSHOT for the RESTORE phase. This process is because SNAPSHOT does not change the CA/CI splits.

REORGANIZE provides (optionally) for the rebuild of alternate indexes (see the “AIX Keyword” on page 442).

If you require more information about the BACKUP or RESTORE process used by the REORGANIZE function, refer to Chapter 7, “Working with Backups” and Chapter 10, “Restoring Data Sets”.
Using the RUV REORGANIZE Function

To use the RUV REORGANIZE function, perform the following steps:

1. Determine which VSAM files to REORGANIZE.

   The REORGANIZE function can select individual VSAM files, or it can process all of the files in a VSAM group that you have defined with the ADD or UPDATE VSAM_GROUP_DEFINITION command (as described in “Defining Groups” on page 601).

2. Code the JCL to execute the REORGANIZE function of the RUVZSM0 utility.

   The JCL statements for executing the REORGANIZE function of the RUVZSM0 utility differ depending on whether you request a REORGANIZE for individual VSAM files or for a defined VSAM group. To keep the explanations simpler, this manual describes the JCL for individual file REORGANIZEs separately from the JCL for group REORGANIZEs. You can only code one file or one group per RUV control card; however, you can have multiple control cards per step.

   - Request REORGANIZE for individual VSAM files with the REORGANIZE VSAM_IN command.

   - Request REORGANIZEs for a defined VSAM group with the REORGANIZE VSAM_GROUP command.

3. Submit the JCL for execution.

4. Check the results of the execution.
Reorganizing Individual VSAM Files

Reorganize a VSAM file using the REORGANIZE VSAM_IN command, as shown in the following example:

```
REORGANIZE VSAM_IN(GNL.LEDGER)
BACKUP_OUT(BKUP01A LOCATION("VAULT") STATUS(ACTIVE))
AIX(YES)
RETAIN_BACKUP(YES)
```

Input VSAM Files

For reorganization of individual VSAM files, RUV can dynamically allocate the input VSAM files to be copied. You do not need to provide DD statements for these files if you specify the data set name as the value of the VSAM_IN keyword on the REORGANIZE VSAM_IN command. If you provide a DD statement for the input VSAM file, specify the ddname of the DD statement as the value of the VSAM_IN keyword.

REORGANIZE VSAM_IN Command

To REORGANIZE an individual file, code one REORGANIZE VSAM_IN command and its related keywords in the SYSIN data set. You can include multiple REORGANIZE VSAM_IN commands to reorganize multiple individual VSAM files.

Keywords on the REORGANIZE VSAM_IN command identify the file to reorganize and the options to use for the reorganization process. The following sections describe the valid keywords in the order that they are listed.

VSAM_IN Keyword

Use the required VSAM_IN keyword on the REORGANIZE VSAM_IN command to identify the individual file to reorganize. You can code only one VSAM_IN keyword on each REORGANIZE VSAM_IN command. Set the value to the ddname of the DD statement that describes the input VSAM data set in the JCL, or set it to the fully qualified data set name of the VSAM file to have RUV dynamically allocate the data set. Wild cards are not allowed.
INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the REORGANIZE VSAM_IN command to produce a report of the resources without actual execution of the reorganize process. Because no execution occurs, no data sets are cataloged if you use this option.

Set one of the following values for the INVENTORY_ONLY keyword.

The default value is NO.

NO  Reorganize without a preliminary resource report.

YES  Produce a preliminary report of VSAM and reorganize file resources, but do not perform the reorganize process.

BACKUP_OUT Keyword

Use the required BACKUP_OUT keyword on the REORGANIZE VSAM_IN command to identify the data set to contain the backup of the VSAM data set. The backup contains a copy of all records from the input VSAM file, along with the control information about the file. At completion of the backup process, RUV registers the information about the backup file in the repository. Depending on the value of the RETAIN_BACKUP parameter, this backup might be retained or deleted after the reorganize is complete.

Set the value of the BACKUP_OUT keyword to the ddname that identifies the output data set in the JCL. For more information about coding the DD statement, see “Output Backup Data Sets” on page 291.

Code these keywords and their values after the ddname value but before the closing parenthesis for the BACKUP_OUT value.

Other Optional Keywords

You can use the following optional keywords with the REORGANIZE VSAM_IN command, the BACKUP_OUT keyword, or the OUTPUT_MODEL keyword to set options for handling the backup data set:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)
AIX Keyword

Use the optional AIX keyword with the VSAM_IN keyword on the REORGANIZE VSAM_IN command to specify whether to rebuild the alternate indexes.

Set one of the following values for the AIX keyword. The default value is YES.

YES Rebuild alternate indexes.

NO Suppress the index build process.

RETAIN_BACKUP Keyword

Use the optional RETAIN_BACKUP keyword with the VSAM_IN keyword on the REORGANIZE VSAM_IN command to specify whether to keep or delete the backup after the reorganization is complete.

Set one of the following values. The default value is NO.

NO Delete and scratch the backup after the reorganization is complete.

YES Retain the backup after the reorganization is complete for all normal RUV functions (RESTORE or RECOVER).

VSAM_NAME_MODEL Keyword

Use the VSAM_NAME_MODEL keyword on the REORGANIZE VSAM_IN command or the REORGANIZE VSAM_GROUP command to create an Instant Snapshot copy and register it as a backup in the RUV repository.

The VSAM_NAME_MODEL keyword also specifies the name of a pre-existing VSAM name model that you want RUV to use to construct the data set name of the copy. SUF controls all other characteristics of the copy.

Specify a single specific user-defined name (1-32 characters) as the value of the VSAM_NAME_MODEL keyword value.
Reorganizing VSAM Groups

Figure 129 shows model JCL for executing the REORGANIZE function for VSAM groups, as specified with the REORGANIZE VSAM_GROUP command.

NOTE
Before the REORGANIZE VSAM_GROUP command is executed, the VSAM group must be defined with the ADD VSAM_GROUP_DEFINITION or UPDATE VSAM_GROUP_DEFINITION command.

Figure 129 REORGANIZE VSAM_GROUP Model JCL

```
//jobname JOB statement
//BACKUP EXEC PGM=RUVZSM0
//STEPLIB DD DISP=SHR,DSN=yourname.LOAD
//SYSPRINT DD SYSOUT=*  
//SYSLIB DD *
/*
* ASSUME THAT VSAM GROUP AND OUTPUT MODEL ARE ALREADY DEFINED
* OPTIONAL SET COMMAND TO CONTROL ENVIRONMENT OPTIONS
SET keywords;
* REQUIRED BACKUP COMMAND TO INVOKE BACKUP FUNCTION FOR GROUP
REORGANIZE VSAM_GROUP(name32)
  INVENTORY_ONLY(NO | YES) } }
OUTPUT_MODEL(name32)
  LOCATION(locname8)
  COMMENT(comment_fields) }
  STATUS(ACTIVE | INACTIVE) }
) )
; ;
/*
```

The following information applies specifically to RUVZSM0 job steps that contain the REORGANIZE VSAM_GROUP command. For information about the standard DD statements and control statements that are common to all RUVZSM0 jobs, see “Using the RUV Batch Interface” on page 79.

Input VSAM Files

For a VSAM group, RUV always dynamically allocates the input VSAM files to be copied. The data set names are identified in the VSAM group definition. In the backup JCL, you do not provide DD statements for these files.
Output Backup Data Sets

For VSAM groups, RUV dynamically allocates the output data set for each requested backup copy from the information that you specify with the OUTPUT_MODEL keyword on the REORGANIZE VSAM_GROUP command. For more information, see “OUTPUT_MODEL Keyword” on page 297.

REORGANIZE VSAM_GROUP Command

To REORGANIZE a defined group of VSAM files, code one REORGANIZE VSAM_GROUP command and its related keywords in the SYSIN data set. You can include multiple REORGANIZE VSAM_GROUP commands to create backups for multiple VSAM groups.

Keywords on the REORGANIZE VSAM_GROUP command identify the group to be reorganized and the options to use for the reorganize process. The following sections describe the valid keywords in the order that they are listed in the model JCL in Figure 129.

VSAM_GROUP Keyword

Use the required VSAM_GROUP keyword on the REORGANIZE VSAM_GROUP command to identify the VSAM group to reorganize. You can code only one VSAM_GROUP keyword on each REORGANIZE VSAM_GROUP command. Set the value to the name of the previously-defined VSAM group.

For more information about defining VSAM groups, see “Defining Groups” on page 601.

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the REORGANIZE VSAM_GROUP command to produce a report of the resources without actual execution of the reorganize process. Because no execution occurs, no data sets are cataloged if you use this option.

Set one of the following values for the INVENTORYONLY keyword.

The default value is NO.

NO Reorganize without a preliminary resource report.

YES Produce a preliminary report of VSAM and backup file resources, but do not perform the reorganize process.
**OUTPUT_MODEL Keyword**

Use the OUTPUT_MODEL keyword on the REORGANIZE VSAM_GROUP command to identify the output model that contains the allocation information that you want to use for the output backup data set. The keyword value is the name of a previously defined output model.

You can define an output model in the same job step that executes the reorganize function, but typically you would define output models at the same time that you define VSAM groups. For more information, see “Using the ADD OUTPUT_MODEL Command” on page 615.

**Other Optional Keywords**

You can use the following optional keywords with the OUTPUT_MODEL keyword to set options for handling the backup data set:

- LOCATION (see “Using the LOCATION Keyword” on page 110)
- STATUS (see “Using the STATUS Keyword” on page 112)
- COMMENT (see “Using the COMMENT Keyword” on page 107)

**AIX Keyword**

Use the optional AIX keyword on the REORGANIZE VSAM_GROUP command to specify whether to rebuild the alternate indexes.

Set one of the following values for the AIX keyword. The default value is **YES**.

- **YES**  Rebuild alternate indexes.
- **NO**  Suppress the index build process.

**RETAIN_BACKUP Keyword**

Use the optional RETAIN_BACKUP keyword on the REORGANIZE VSAM_GROUP command to specify whether to keep or delete the backup after the reorganization is complete.

Set one of the following values. The default value is **NO**.

- **NO**  Delete and scratch the backup after the reorganization is complete.
- **YES**  Retain the backup after the reorganization is complete for all normal RUV functions (RESTORE or RECOVER).
**VSAM_NAME_MODEL Keyword**

Use the VSAM_NAME_MODEL keyword on the REORGANIZE VSAM_IN command or the REORGANIZE VSAM_GROUP command to create an Instant Snapshot copy and register it as a backup in the RUV repository.

The VSAM_NAME_MODEL keyword also specifies the name of a pre-existing VSAM name model that you want RUV to use to construct the data set name of the copy. SUF controls all other characteristics of the copy.

Specify a single specific user-defined name (1-32 characters) as the value of the VSAM_NAME_MODEL keyword value.
Restoring Data Sets

This chapter discusses how to use RUV to restore data sets. It also describes how to use non-RUV restore processes with RUV. The following information is included:

Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 448
RUV Actions for Restore . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 448
Automatic Restore during Forward Recovery . . . . . . . . . . . . . . . . . . . . . . . . . . . 448
NEW NAME Restore . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 449
RUV CURRENT BACKUP Restore . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 449
Point-in-Time Restore . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 449
Non-RUV Backups . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 450
Alternate Indexes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 450
Number of Reader Tasks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 450
SMS Backups . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 451
IDCAMS Delete/Define Operations for VSAM Files . . . . . . . . . . . . . . . . . . . . . . . . 451
Restoring a Backup-While-Open Backup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 451
Forward Recovery and the RBA0 Backup Detection Feature . . . . . . . . . . . . . . . . . . . 452
Summary of Forward Recovery Actions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 452
Current Backup Specified . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 454
Backup Data Set Name Specified . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 454
Recovery Time Specified with No Archives or Backups Specified . . . . . . . . . . 454
Recovery Time Specified with No Archives or Backups Specified . . . . . . . . . 454
Recovery Time and Archives Specified with No Backups Specified . . . . . . . . 455
Archives Specified with No Recovery Time or Backups Specified . . . . . . . . . 456
Restoring Files from Backups . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 457
Running the Restore Job . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 457
Using the RESTORE VSAM_FILE Command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 457
Using the RESTORE SEQ_FILE Command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 458
Using the VSAM_FILE Keyword . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 458
Using the SEQ_FILE Keyword . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 459
Using the NEW_NAME KEYWORD . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 459
Using the START_TIME keyword . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 460
Using the STOP_TIME keyword . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 460
Using the AIX Keyword for VSAM Files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 460
Using the BACKUP_IN Keyword . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 461
Coding RESTORE VSAM_FILE Statements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 462
Coding RESTORE SEQ_FILE Statements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 462
Overview

Restoring VSAM files rebuilds the VSAM cluster from a backup. You would perform a restore when the current backup is the point or condition you want to establish for the file.

The following sections provide general information about restoring files from backups.

RUV Actions for Restore

When the RESTORE command contains the name of a specific ddn8 or dsn44 backup, RUV uses the specified backup. If the backup does not exist or is corrupted, the recovery fails.

When the RESTORE command contains a CURRENT_BACKUP keyword, RUV takes one of the actions listed in Table 16.

Table 16  RUV Actions for Restore with CURRENT_BACKUP Specified

<table>
<thead>
<tr>
<th>Active RUV Backup Exists</th>
<th>SMS Backup Exists</th>
<th>Action to be Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>restore RUV backup</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>request SMS restore</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>restore the most current backup based on date stamp</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>error</td>
</tr>
</tbody>
</table>

RUV determines how a VSAM file is defined by using the information registered in the backup file or in the repository. This information allows RUV to redefine VSAM files even when they do not exist. Sequential files must be defined by the customer; RUV will not define sequential files.

Automatic Restore during Forward Recovery

The forward recovery process automatically invokes the restore process if a VSAM data set does not exist in the z/OS catalog. The most current RUV or SMS backup is selected.
**NEW NAME Restore**

RUV can restore a file to a different name. You might want to use this option to test and audit your RUV restoration and recovery environment prior to actual production. If the VSAM file has paths, you should restore only the base and then rebuild the paths. A restore of an SMS backup with paths results in the base cluster being renamed but the path names having the old names.

**RUV CURRENT BACKUP Restore**

RUV will restore a VSAM file automatically during a recovery if you specify the BACKUP_IN(CURRENT_BACKUP) keyword default for restore, or if the VSAM file does not exist.

**Point-in-Time Restore**

RUV can restore any backup that it has taken. You may need to restore a file to a particular historical point in time. To determine the correct backup to use to restore the data to the point in time that you want, run the REPORT BACKUP_FILE command or the REPORT VSAM_FILE or REPORT SEQ_FILE command at the REPORT(DETAIL) level.

**NOTE**

Use of the REPORT VSAM_FILE or REPORT SEQ_FILE commands and REPORT(DETAIL) keywords with wildcards might run for a long time, depending on the number of actual records.
Non-RUV Backups

RUV can restore a non-RUV backup (a backup created with a backup process other than RUV) automatically through support for external backups. See “Using SMS Backups and IDCAMS REPRO Copies” on page 314 and “Creating External Backups” on page 315.

You can use the BACKUP_METHOD(NO_EXTERNAL) keyword on the SET or SET DEFAULT command to prevent RUV from attempting to restore an external backup automatically.

When the non-RUV backup and the file are not registered to RUV, you cannot use RUV for automatic restore processing. You must manually specify the data set with the RESTORE BACKUP_IN keyword. See “Using the RESTORE VSAM_FILE Command” on page 457 and “Using the RESTORE SEQ_FILE Command” on page 458.

Alternate Indexes

RUV can rebuild alternate indexes after a successful restore of a VSAM file.

Number of Reader Tasks

Depending on the number of files and the media type (disk or tape), RUV initiates multiple restore and index-build tasks. You can use the READER_TASK option to control the maximum number of tasks that RUV should run in parallel. Limiting the number of tasks also limits the maximum number of tape drives in use at one time.
**SMS Backups**

RUU will search the repository for RUV backups, followed by a search of SMS backups, and then decide on which is the most current backup.

An SMS backup is reported as follows:

```
SMS_BKUP.D2009254.T1048446 2009.254 10:48:44       SMS
```

**IDCAMS Delete/Define Operations for VSAM Files**

You can perform the IDCAMS delete/define operations in separate steps from the RUV restore process, or you can use the ENTER_IDCAMS and EXIT_IDCAMS commands to perform the delete/define operations in the same job step with the restore. For more information, see “Invoking IDCAMS from an RUV Job Step” on page 514.

In addition, RUV can create IDCAMS DELETE/DEFINE statements for a VSAM file from the information that is stored in the system catalog or in the RUV repository. For more information, see “Delete/Define Statements in ISPF Edit Panel” on page 385.

**Restoring a Backup-While-Open Backup**

The restore of a Backup-While-Open backup to a VSAM file results in RUV issuing the following informational message:

```
RUV202732I RUV A fuzzy backup was restored into the VSAM DSName dsname.
```

In this case, you should try to recover the data set instead of restoring it. The backup provides a starting point for the recovery of the data. The recovery would capture all the archive data associated with the updates being done while the backup was taken. After the recovery, the data would be complete and accurate.
Forward Recovery and the RBA0 Backup Detection Feature

You can recover a VSAM file across RBA0 backups. The following sections describe the actions that RUV takes during forward recovery if you are using the RBA0 backup detection feature. The actions depend on the keywords that you specify on the RECOVER FORWARD command.

If an RBA0 backup is selected for restore or recovery, the VSAM file is deleted and defined (making the file empty), and any archive records that were written since the backup are applied.

Summary of Forward Recovery Actions

Figure 130 summarizes the actions that RUV takes during forward recovery. If you do not specify the START_TIME keyword, RUV initially sets the start time to the start time of the earliest record. If you do not specify the STOP_TIME keyword, RUV initially sets the stop time to the current time.
Figure 130  Summary of Forward Recovery Actions

Start

Is restore needed?

Yes

(You have specified BACKUP_IN or the VSAM file does not exist.)

No

(You did not specify BACKUP_IN and the VSAM file exists.)

Is ARCHIVE_IN specified?

Yes

(You have specified START_TIME or RUV resets the start time to the archive start time.)

Are the specified archives in the repository?

Yes

RUV processes repository archive records to determine the start time and stop time.

No

Did you specify START_TIME?

Yes

Error: You must specify START_TIME or ARCHIVE_IN.

No

Did an RBA0 backup occur between the start time and stop time?

Yes

RUV applies archive records that fall between the start time and stop time.

No

Stop

Is BACKUP_IN specified?

Yes

RUV restores the most recent RBA0 backup.

No

Is a more recent RBA0 backup available

Yes

RUV resets the start time to the time of the applied backup.

No

RUV restores the specified backup.

No

A
Current Backup Specified

If the RECOVER FORWARD command specifies the current (latest or most recent) backup with the default BACKUP_IN(CURRENT_BACKUP) keyword, RUV always selects the latest backup, regardless of whether it is an RBA0 backup or other type of backup.

Backup Data Set Name Specified

If the RECOVER FORWARD command specifies the data set name of a backup with the BACKUP_IN(dsn44) keyword, RUV takes action as follows:

- If no RBA0 backup has occurred in the time frame after the specified backup and before the target recovery time, RUV selects the specified backup.
- If one or more RBA0 backups have occurred in the time frame after the specified backup and before the target recovery time, RUV selects the latest RBA0 backup that has occurred within that time frame.

Recovery Time Specified with No Archives or Backups Specified

If the RECOVER FORWARD command specifies a target recovery time frame (with the START_TIME keyword, the STOP_TIME keyword, or both), but it does not specify an input archive (with the ARCHIVE_IN keyword) or an input backup (with the BACKUP_IN keyword), RUV takes action as follows:

- If the data set to be recovered does not exist, RUV selects the latest backup that has occurred within the specified time frame. It adjusts the START_TIME value to the time of the selected backup. RUV performs a delete/define of the VSAM file from the information that is stored in the repository, and then performs the restore or recovery.
- If the data set to be recovered exists, RUV selects the latest backup that has occurred within that time frame. If an RBA0 backup is selected, RUV performs a delete/define operation for the file and uses operating system catalog information. RUV adjusts the START_TIME value to the time of the backup.
Recovery Time and Archives Specified with No Backups Specified

If the RECOVER FORWARD command specifies a target recovery time frame (with the START_TIME keyword, the STOP_TIME keyword, or both), and it specifies one or more input archives (with the ARCHIVE_IN keyword), but it does not specify an input backup (with the BACKUP_IN keyword), RUV takes action as follows:

- If the data set to be recovered does not exist, RUV selects the latest backup (regardless of whether it is an RBA0 backup or other type of backup) that has occurred within the specified time frame. It adjusts the START_TIME value to the time of the selected backup, restores the file from the backup, and applies archive records (from the specified archives) that have times within the adjusted time frame.

- If the data set to be recovered exists but the archives are not registered in the repository, RUV does not attempt to select and restore any type of backup. It applies archive records (from the specified archives) that have times within the specified time frame.

- If the data set to be recovered exists and the archives are registered, RUV examines the specified archives to find the start time and stop time of each data set to be recovered (as specified with the VSAM_FILE keyword). It adjusts the START_TIME value and the STOP_TIME value to the start time and stop time from the archives. If one or more RBA0 backups have occurred within this adjusted time frame, RUV ignores any other types of backups and selects the latest RBA0 backup that occurred during this time frame. RUV adjusts the START_TIME value to the time of the RBA0 backup and restores the file from the RBA0 backup. Then RUV applies archive records (from the specified archives) that have times within the readjusted time frame. It also issues an informational message with condition code 0.
If the RECOVER FORWARD command specifies one or more input archives (with the ARCHIVE_IN keyword), but it does not specify a target recovery time (with the START_TIME keyword, the STOP_TIME keyword, or both) or an input backup (with the BACKUP_IN keyword), RUV takes action as follows:

- If the data set to be recovered does not exist, RUV selects the latest backup (regardless of whether it is an RBA0 backup or other type of backup). RUV adjusts the START_TIME value to the time of the selected backup, restores the file from the backup, and applies archive records (from the specified archives) that have times within the adjusted time frame.

- If the data set to be recovered exists but the archives are not registered in the repository, RUV does not attempt to select any type of backup. It applies all archive records (from the specified archives).

- If the data set to be recovered exists and the archives are registered, RUV examines the specified archives to find the start time and stop time of each data set to be recovered (as specified with the VSAM_FILE keyword). It adjusts the START_TIME value and the STOP_TIME value to the start time and stop time from the archives. If one or more RBA0 backups have occurred within this adjusted time frame, RUV ignores any other types of backups and selects the latest RBA0 backup that occurred during this time frame. RUV adjusts the START_TIME value to the time of the RBA0 backup and restores the file from the RBA0 backup. Then RUV applies archive records (from the specified archives) that have times within the readjusted time frame. It also issues an informational message with condition code 0.
Restoring Files from Backups

The following sections provide information for restoring a file from a backup.

Running the Restore Job

To restore files, perform the following procedure:

1. Determine which VSAM or sequential file or files to restore.
2. Build the JCL for the restore job.
3. Build the RESTORE VSAM_FILE or RESTORE SEQ_FILE command for each file you want to restore.
4. Run the restore job.
5. Review results of the restore job.

Using the RESTORE VSAM_FILE Command

Restore a VSAM file using the RESTORE VSAM_FILE command, as shown in the following example:

```
RESTORE VSAM_FILE(GNL.LEDGER
    NEW_NAME(PAY2.GNL)
    AIX(YES)
    BACKUP_IN(CURRENT_BACKUP)
)
VSAM_FILE(GNL.PAYROLL)
    AIX(NO)
    BACKUP_IN(CURRENT_BACKUP)
)
```
Using the RESTORE SEQ_FILE Command

Restore a sequential file using the RESTORE SEQ_FILE command, as shown in the following example:

```
RESTORE SEQ_FILE(GNL.LEDGER
   NEW_NAME(PAY2.GNL)
   BACKUP_IN(CURRENT_BACKUP)
 )
SEQ_FILE(GNL.PAYROLL)
   BACKUP_IN(CURRENT_BACKUP)
 )
;```

Using the VSAM_FILE Keyword

Use the required VSAM_FILE keyword on the RESTORE VSAM_FILE command to identify a VSAM file to restore. You can code multiple VSAM_FILE keywords on each command. Set the value to the fully qualified data set name of the VSAM file to restore.

If no BACKUP_IN keyword is specified, the RESTORE VSAM_FILE command will use the value of CURRENT_BACKUP as the default backup source.

You can code the following keywords with the VSAM_FILE keyword to set options for handling this VSAM data set:

- NEW_NAME
- AIX
- BACKUP_IN

Code these keywords and their values after the data set name value but before the closing parenthesis for the VSAM_FILE value.
Using the SEQ_FILE Keyword

Use the required SEQ_FILE keyword on the RESTORE SEQ_FILE command to identify a sequential file to restore. You can code multiple SEQ_FILE keywords on each command. Set the value to the fully qualified data set name of the file to restore.

If no BACKUP_IN keyword is specified, the RESTORE SEQ_FILE command will use the value of CURRENT_BACKUP as the default backup source.

You can code the following keywords with the SEQ_FILE keyword to set options for handling this sequential data set:

- NEW_NAME
- BACKUP_IN

Code these keywords and their values after the data set name value but before the closing parenthesis for the SEQ_FILE value.

Using the NEW_NAME KEYWORD

Use the optional NEW_NAME keyword to restore a file to a new name. With an alternate data set name, you can copy production files to test and audit production environments. You can perform many recoveries against the file without risking damage to an actual production file.

The NEW_NAME value restores a data set to a different VSAM cluster or sequential file. Only the base cluster is restored for VSAM. You must rebuild alternate paths manually to recreate a complete data set structure.

If you build a NEW_NAME VSAM data set before the restore actions are performed, and that data set has alternate indexes with the data set, you can then use the AIX keyword to have RUV rebuild alternate indexes for that NEW_NAME VSAM data set.

If you do not build the NEW_NAME VSAM data set before the restore actions are performed, but you specify NEW_NAME VSAM data sets as part of the initial restore action, RUV will not rebuild alternate indexes for that NEW_NAME VSAM data set.

---

**WARNING**

Do not use the NEW_NAME feature with SMS backups that have paths. Use SMS facilities to restore and rename those data sets.
Using the START_TIME keyword

The START_TIME keyword is valid with the RESTORE VSAM_FILE or RESTORE SEQ_FILE command. You can use this keyword with the STOP_TIME keyword to specify the backup data to restore.

Use the optional START_TIME keyword to specify the start (earliest) timestamp of the range of data to include in the restore process. This timestamp is required if you are using a backup that was not produced by RUV. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the START_TIME Keyword” on page 111.

Using the STOP_TIME keyword

The STOP_TIME keyword is valid with the RESTORE VSAM_FILE or RESTORE SEQ_FILE command. You can use this keyword with the START_TIME keyword to select the range of records to print from the archive.

Use the STOP_TIME keyword to specify the stop (latest) timestamp of the data to include in the restore process. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the RUV ISPF Interface” on page 113.

Using the AIX Keyword for VSAM Files

Use the optional AIX keyword with the VSAM_FILE keyword on the RESTORE VSAM_FILE command to specify whether to rebuild the alternate indexes. Set one of the following values. The default value is YES.

YES  Rebuild alternate indexes.

NO   Suppress the index build process.
Using the BACKUP_IN Keyword

Use the optional BACKUP_IN keyword on the RESTORE VSAM_FILE or RESTORE SEQ_FILE command to identify the backup data set to use as input to the restore process.

You can allow RUV to choose the latest active backup based on information in the repository by setting the value of the BACKUP_IN keyword to CURRENT_BACKUP. The CURRENT_BACKUP value causes RUV to look up the most current active backup automatically and perform the delete/define, if needed, against the VSAM file before restoration.

If no keyword is specified, the RESTORE VSAM_FILE or RESTORE SEQ_FILE command uses the value of CURRENT_BACKUP as the default backup source.

When you specify a backup rather than allow RUV to choose the backup based on information in the repository, you are implicitly requesting a point-in-time recovery. The point in time is the GMT timestamp (the start time) of the backup file.

To restore a specific backup file, set the value of the BACKUP_IN keyword to the fully qualified data set name of the backup file (RUV will dynamically allocate the data set).

**NOTE** If the RESTORE VSAM_FILE or RESTORE SEQ_FILE command includes the BACKUP_IN keyword, the restore report contains details about the backup used in the RESTORE command.
Coding RESTORE VSAM_FILE Statements

To code a RESTORE VSAM_FILE statement, use the following syntax:

```plaintext
RESTORE VSAM_FILE(dsn44)
   {NEW_NAME(dsn44)}
   {AIX(YES|NO)}
   {BACKUP_IN
       (CURRENT_BACKUP)|(dsn44)}
   { INVENTORY_ONLY(NO|YES) }
   { VSAM_FILE(dsn ... ) }
;
```

Coding RESTORE SEQ_FILE Statements

To code a RESTORE SEQ_FILE statement, use the following syntax:

```plaintext
RESTORE SEQ_FILE(dsn44)
   {NEW_NAME(dsn44)}
   {BACKUP_IN
       (CURRENT_BACKUP)|(dsn44)}
   { INVENTORY_ONLY(NO|YES) }
   { SEQ_FILE(dsn ... ) }
;
```
Recovering VSAM Data Sets

This chapter describes how to use RUV for recovery of VSAM data sets in a local system. The following information is included:

- Overview of Recovery on a Local System
- Forward Recovery
- Unit-of-Work Recovery
- Backout Recovery
- Sort Parameters
- Using RUV for Forward Recovery
  - Running an RUV Forward Recovery
  - Using the RECOVER FORWARD Command
  - Coding RECOVER FORWARD Statements
  - Using the AIX Keyword
  - Using the ARCHIVE_IN Keyword
  - Using the BACKUP_IN Keyword
  - Using the BUFND Keyword
  - Using the BUFNI Keyword
  - Using the INVENTORY_ONLY Keyword
  - Using the LSR_POOL Keyword
  - Using the NEW_NAME Keyword
  - Using the RECOVERY_MODE Keyword
  - Using the SELECTION_EXIT Keyword
  - Using the START_TIME Keyword
  - Using the STOP_TIME Keyword
  - Using the SUPPRESS_RECOVERY_ERROR Keyword
  - Using the VSAM_FILE Keyword
- Using RUV for UOW Recovery
  - Running an RUV UOW Recovery
  - Using the RECOVER COMPLETED_UNIT_OF_WORK Command
  - Coding the RECOVER COMPLETED_UNIT_OF_WORK Statements
  - Using the ARCHIVE_IN Keyword
  - Using the BACKUP_IN Keyword
  - Using the INVENTORY_ONLY Keyword
  - Using the REPORT UNIT_OF_WORK Command
  - Using the START_TIME Keyword
  - Using the STOP_TIME Keyword
  - Using the UNIT_OF_WORK_IN Keyword
Using the UNIT_OF_WORK_OUT Keyword ........................................... 487
Using the VSAM_FILE Keyword ................................................. 488
Using the VSAM_GROUP Keyword ................................................ 488
Using RUV for Backout Recovery .................................................. 489
Performing an RUV Backout Recovery ........................................ 489
Using the RECOVER BACKOUT Command .................................... 489
Coding the RECOVER BACKOUT Statements .................................. 490
Using the ARCHIVE_IN Keyword ............................................... 491
Using the BUFND Keyword ......................................................... 491
Using the BUFINI Keyword ......................................................... 491
Using the INVENTORY_ONLY Keyword ........................................ 492
Using the LSR_POOL Keyword ..................................................... 492
Using the RECOVERY_MODE Keyword ....................................... 492
Using the SELECTION_EXIT Keyword ....................................... 493
Using the START_TIME Keyword .............................................. 493
Using the STOP_TIME Keyword ............................................... 493
Using the VSAM_FILE Keyword ................................................... 494
Using the ISPF Interface for Recovery ........................................ 495
Accessing the VSAM Sphere Recovery Panels ............................ 495
Using the VSAM Sphere Recovery - Control Panel ....................... 496
Using the VSAM Sphere Recovery - Filters Panel ......................... 498
Using the VSAM Sphere Recovery - Group Names Panel ............... 499
Using the VSAM Sphere Recovery - VSAM File Names Panel .......... 500
Specify a Recovery Time Range Panel ........................................ 501
Select Backup Files Panel ......................................................... 503
Select Forward or Backout Recovery Panel ................................. 505
Select New Names Panel ........................................................... 506
Generate JCL Panel ................................................................. 508
Save Recovery JCL Panel ......................................................... 510
Submit JCL Panel ................................................................. 511
Understanding RUV Exception Reasons ...................................... 512
Invoking IDCAMS from an RUV Job Step .................................... 514
Using the IDCAMS Commands .................................................. 514
Coding the ENTER_IDCAMS and EXIT_IDCAMS Commands ....... 514
Introducing a Delay into an RUV Process .................................... 515
Coding the TEST DELAY Command .......................................... 515
Using the TEST DELAY Command ............................................ 515
Using the SELECTION_EXIT User Exit ....................................... 517
User Exit Calls ................................................................. 517
User Exit Return Codes .......................................................... 520
User Exit Parameter List .......................................................... 520
Overview of Recovery on a Local System

RUV recovers the entire VSAM sphere, including the base cluster and alternate indexes. The recovery process includes forward recovery and backout recovery.

The following sections provide general information about local recovery topics.

Forward Recovery

If a DASD error has caused the loss of a VSAM file or made the file unreadable, you can use the forward recovery process to rebuild the lost or damaged file. You can also use forward recovery to recover a file forward to a point in time, such as the completion of a particular job. The forward recovery process uses the after-image journal records to rebuild the file and will use a backup as required.

The RUV Batch Journaling Facility can perform backout recovery automatically if a program abend occurs. For more information, see Chapter 4, “Using the Batch Journaling Facility.”

New Data Set Name Recovery

RUV can forward recover a VSAM file to a new data set name. You can test and audit your RUV restoration and recovery environment prior to actual production.

Automated Forward Recovery

If you use RUV to perform backups, RUV can automate every element of the recovery process, including restoring the backups and determining the start timestamps (if it does a restore) and stop timestamps of the range of records to use for the recovery.

The normal construction of the start and stop timestamps is internal to RUV. A backup (if used, as registered in the repository) contains and controls the start time needed for selecting the correct archive files. Because RUV manages all backups and archives in the repository, for a typical recovery you do not supply a stop time, but you must supply a backup or a start time.

RUV forward recovers a VSAM file automatically if you specify the BACKUP_IN(CURRENT_BACKUP) keyword.
Manual Forward Recovery

If you use a product other than RUV or SMS to perform backups, you must manually restore the backup file and provide RUV with the start time of the backup. RUV may automatically restore RUV backups during a recovery. RUV automatically picks the start time to select the necessary archive files for the recovery process if it restores a backup. Alternatively, you can provide the DSNs of the archive files you want to recover.

You can also supply start and stop timestamps to recover a file to a particular point in time, such as to synchronize with a database management system.

Start Timestamp Generation

If you do not provide a start timestamp, RUV generates one from data in the repository or sets the value to the default value, which is low values (X'0000000000000000').

To generate the timestamp from data in the repository, RUV finds the earliest timestamp for a forward recovery record associated with a specific VSAM file and uses its timestamp. However, if the forward recovery process needs to restore a backup, RUV resets the specified start time to the start time of the restored backup (to preserve data integrity).

If you need to use a particular time range or want to override any RUV time-range calculations, first restore the file manually and then perform the recovery.

Stop Timestamp Generation

If you do not provide a stop timestamp, RUV recovers the file to the current time. If you provide a stop timestamp for the recovery, RUV bypasses its normal process of constructing a stop timestamp.

Forward Recovery with Record Level Sharing

RUV can perform forward recovery of VSAM files that use the VSAM Record Level Sharing (RLS) feature. During the recovery operation, RUV can interact with the sharing control data set (SHCDS) to release (and later reapply) the retained locks that are held for individual records. To enable this function, use the RLS_RECOVER_PROTECTION keyword and establish UPDATE authority with your security package (such as RACF). For details, see “Using the RLS_RECOVER_PROTECTION Keyword” on page 135.
IDCAMS Delete/Define Operations

You can perform the IDCAMS delete/define operations in separate steps from the RUV restore process, or you can use the ENTER_IDCAMS and EXIT_IDCAMS commands to perform the delete/define operations in the same job step with the restore. For more information, see “Invoking IDCAMS from an RUV Job Step” on page 514.

RUV Actions for Forward Recovery

The actions that RUV takes for a forward recovery depend on several conditions. See Table 17.

<table>
<thead>
<tr>
<th>Conditions Specified</th>
<th>VSAM File</th>
<th>Does Not Exist</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Archive</strong></td>
<td><strong>Backup</strong></td>
<td><strong>Exists</strong></td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>use specified archive / backup</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete / define / restore</td>
</tr>
<tr>
<td></td>
<td></td>
<td>recover</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>you restore VSAM file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>use specified archive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>recover</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>use specified backup</td>
</tr>
<tr>
<td></td>
<td></td>
<td>use archives selected from repository, using the time of the backup as START_TIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete / define / restore / recover</td>
</tr>
<tr>
<td></td>
<td></td>
<td>build AIXs* if required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>if no archives exist, this is a restore with no recovery</td>
</tr>
</tbody>
</table>
A unit-of-work (UOW) recovery operation is a forward recovery that applies only completed transactions to the backup. The UOW recovery operation uses a backup of a data set and all after-image journal records that were created from the time the backup was made until the stop time specified for recovery. Updates that are part of an incomplete UOW are not applied to the current recovery. Since adds to ESDS files are always applied by CICS, RUV also applies the ESDS adds during the recovery.

Data on the incomplete transactions can be written to a separate output file. It can be printed and viewed for analysis with the PRINT ARCHIVE_IN(dsn) command. This data can be used as input for a subsequent UOW recovery.

**Minimum UOW Requirements**

To perform a UOW recovery, the archives of both the user journal and the DFHLOG for the time range being recovered must be available and registered.

As part of your normal procedures, when using Transaction Server you should use CEMT command to flush the buffers before you issue the ARCHIVE command to create the archives. See the IBM CICS documentation for the CEMT SET JOU(DFHJ#) FLUSH command.

When choosing a stop time for a UOW recovery, the stop time must be included in the archives of both the user journal and DFHLOG.
For example, if the user journal archive to be used includes data activity up to 1:00 PM GMT and the DFHLOG archive to be used includes data activity up to 2:00 PM GMT, the latest possible UOW stop time is 1:00 PM GMT because data activities for that time are available in both archives. If you must use a later stop time (for example 1:30 PM GMT), you must issue the command to create a new DFHLOG archive covering the later stop time and use that new archive.

**Use of the Unit-of-Work Concept**

A UOW is a recoverable sequence of operations that is performed by an application between a known beginning of task and known end of task point. A UOW starts when an update of the first recoverable resource is made by CICS transaction. The UOW ends when a sync point is taken to commit the changes or at the end of task (EOT).

In normal operation, if a CICS abend occurs and an emergency restart is performed, CICS identifies all changes that are not committed and backs them out to preserve data integrity.

By using the UOW concept, RUV provides the same integrity when recovering data, not just at the abend, but at any point in time that you specify.

**A UOW Recovery from a Backup**

The differences in the RUV recovery process using the UOW functionality are illustrated in Figure 131.

**Figure 131  UOW Recovery from a Backup**

The first step in a UOW recovery is to restore a valid backup. However, with UOW, before applying an individual transaction, the UOW archive is used to determine if the transaction is complete at the specified recovery point.

- If it is complete, it is applied to the recovered VSAM file.
- If it is in-flight, the transaction is not applied. Information about the in-flight transaction is written to a file that can be used as input data for the next recovery to ensure data integrity. The information can be printed for auditing and troubleshooting analysis.
Setting Up for a UOW Recovery

To perform a UOW recovery, you must configure the CICS file definition to log specific information about the files. As CICS processes transactions the required information is captured in a log stream or journal. You must perform an initial backup to provide a known starting point for future UOW recoveries.

- For CICS with Transaction Server 1.3 or later, the UOW data is written to DFHLOG which contains before images of updated VSAM files and UOW information. If you plan to do a forward recovery using the RUV UOW processing, you must set the retention period (in days) for log data in the z/OS log stream to a value other than zero.

- For prior versions of CICS, the UOW data is written to the DFHJ01A/B/X journal, which may contain both before-images and after-images of updated VSAM files.

- Depending on which version of CICS is used, the data from DFHLOG or journal must be captured or archived by running the RUV archive command and specifying the UNIT_OF_WORK_OUT(ddn) keyword.

Factors that Affect the Type of Recovery

The decision on which type of backup and recovery to perform is based on the results you want at the end of the process.

RUV is designed to handle a recovery with a minimum amount of user input. Unless you have special needs, it is recommended that you use the RUV backups and allow RUV to automatically select the appropriate backups and archives when a recovery is required.

RUV can determine which backup will provide the least amount of archive processing. When running a recovery with the BACKUP_IN(CURRENT_BACKUP) keyword, RUV uses the most current SMS or RUV backup and the associated archive to recover the data set.

RUV generates an output file of in-flight transactions that were not applied during a recovery. This file can be used as one of the inputs on a subsequent recovery.

RUV allows you to override the defaults and specify all the backup or archive values and start and stop time values. When these overrides are specified, RUV recovers the data set using exactly the input data you specified.
Backout Recovery

You can use RUV to back out (or undo) changes that have been made to a VSAM file. The backout recovery process does not use VSAM backup data sets. Instead, it uses the existing VSAM data set and applies before-image journal records to reverse the changes.

RUV cannot perform a backout for ESDS files with paths. Use a forward recovery process instead.

The RUV Batch Journaling Facility can perform backout recovery automatically if a program abend occurs. For more information, see Chapter 4, “Using the Batch Journaling Facility.”

RUV Actions for Backout Recovery

The actions that RUV takes for a backout recovery depend on several conditions. See Table 18.

Table 18  RUV Actions for Backout Recovery

<table>
<thead>
<tr>
<th>Conditions Specified</th>
<th>VSAM File</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Archive</td>
<td>Backup</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTES:

When the command uses the ddname, RUV performs no dynamic allocation. The GDG specification is allowed only in JCL and not within any commands.
If your sort utility defaults prevent RUV from running the recovery, you can place these following parameters in the RUV recovery job stream. These parameters tell the sort utility that it can use all the memory in the region except for 128 KB.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DFSORT/Sort Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>//DFSPARM DD *</td>
<td>DFSORT</td>
</tr>
<tr>
<td>LIST, LISTX, MSGPRT=ALL, SIZE=MAX-128K</td>
<td>DFSORT</td>
</tr>
<tr>
<td>//SORTPARM DD *</td>
<td>Syncsort</td>
</tr>
<tr>
<td>MSG=AP, CORE=MAX-128K, DYNALLOC=ON</td>
<td>Syncsort</td>
</tr>
</tbody>
</table>
Using RUV for Forward Recovery

To run the RUV forward recovery process in batch, execute the RUV utility (program RUVZSM0). For general information about the RUV utility, see “Using the RUV Batch Interface” on page 79.

You can use the ISPF interface to generate recovery JCL. For more information, see “Using the ISPF Interface for Recovery” on page 495.

Running an RUV Forward Recovery

To run an RUV forward recovery, perform the following steps:

1. Determine the VSAM file or files to include for forward recovery.
2. Build the JCL with the correct space allocation.
3. Build the RECOVER FORWARD command to specify the files to recover and the options to use. Specify the setting for incremental backups.
4. Run the recovery job or jobs.

Using the RECOVER FORWARD Command

To perform forward recovery for one or more files in batch, use the RECOVER FORWARD command. All keywords on the command correspond to options you can control.
Coding RECOVER FORWARD Statements

To code a RECOVER FORWARD statement, use the following syntax:

```clike
RECOVER FORWARD
  { INVENTORY_ONLY(NO | YES) }
  VSAM_FILE(dsn44)
    { NEW_NAME(dsn44) }
    { START_TIME(yyyyjjjjhhmmsst) }
    { STOP_TIME(yyyyjjjjhhmmsst) }
    { RECOVERY_MODE(NIS | SIS) }
    { BUFND(nnn) }
    { BUFNI(nnn) }
    { LSR_POOL(YES | NO) }
    { BACKUP_IN
        (CURRENT_BACKUP) | (ddn8_or_dsn44) }
    { SELECTION_EXIT(program8) }
  }
  { VSAM_FILE(dsn44 ... ) }
  { ARCHIVE_IN(ddn8_or_dsn44, ... ) }
;
```

The following examples show a forward recovery of two files in a general ledger application. RUV can invoke a user exit routine during processing of the files.

Example 1

In Example 1, when the CURRENT_BACKUP keyword is used, RUV identifies the input backup and VSAM files for recovery with no additional input from the user.

```clike
RECOVER FORWARD
  VSAM_FILE(GNL.LEDGER
    BACKUP_IN(CURRENT_BACKUP)
    SELECTION_EXIT(GNLEXIT)
  )
  VSAM_FILE(GNL.PAYROLL
  );
```
Example 2

In Example 2, the control statements specify the input backups and VSAM file to use and provide a stop time of 2:00 PM on September 15, 2009 (Julian Day 2009.258).

```
RECOVER FORWARD
   VSAM_FILE(GNL.LEDGER
             STOP_TIME(20092581400000)
             BACKUP_IN(GNL.LEDGER.BKUP)
             SELECTION_EXIT(GNLEXIT)
           )
   VSAM_FILE(GNL.PAYROLL
             STOP_TIME(20092581400000)
             BACKUP_IN(GNL.PAYROLL.BKUP)
             SELECTION_EXIT(GNLEXIT)
           )
   ARCHIVE_IN(GNL.ACCUMARC)
;```

**WARNING**

The LSR_POOL, BUFNI, BUFND, and RECOVERY_MODE parameters can greatly affect the performance of a RECOVER.

Before you change from the defaults, it is strongly recommended that you test the effects of changing these parameters before implementing them in production environments.
Using the AIX Keyword

Use the optional AIX keyword with the RECOVER FORWARD command to rebuild alternate indexes (AIXs) and PATHs that were deleted by the recovery. If YES is selected, any deleted AIXs are rebuilt at the end. If NO is selected, the AIXs are not rebuilt.

If the AIX keyword is used on the RECOVER FORWARD command, it is applied globally to all VSAM files processed. If the AIX keyword is used locally for a single VSAM file, it overrides the global value for that VSAM file only.

Set one of the following values for the AIX keyword. The default value is YES.

YES Rebuild all AIXs and PATHs that were deleted.

NO Do not rebuild the AIXs and PATHs.

NOTE

AIXs that were not deleted are not affected by the AIX keyword. If you specify AIX(NO), an AIX that would not have been deleted otherwise is not deleted during the recovery.

RUV will delete AIXs and PATHs only if the AIX has both UNIQUEKEY and UPGRADE or when a file is restored during a recovery. Otherwise, the AIX is not deleted and the AIX keyword is ignored.
Using the ARCHIVE_IN Keyword

You can use the optional ARCHIVE_IN keyword on the RECOVER FORWARD command to identify an archive data set to use as input to the recovery process.

It is recommended that you let RUV automatically select the archive files for use. When you specify the archive files, you override RUV’s normal selection mechanism. If multiple archive files have active statuses, RUV selects them all for recovery and archive processing, even if they are copies of the same archive file.

If you specify the input archives with the ARCHIVE_IN keyword, you are responsible for providing enough sort work space for sorting the number and size of input archive records. If RUV selects the archives (you omit the ARCHIVE_IN keyword), RUV automatically requests the appropriate amount of sort work space.

To specify the value of the ARCHIVE_IN keyword, use the ddname of the DD statement that describes the archive data set in the JCL, or specify the fully qualified data set names of the archive data sets to have RUV allocate the data set dynamically.

---

**NOTE**

If you specify an ARCHIVE_IN value for an existing data set you may get unexpected results. RUV assumes that you have verified the accuracy of the data set and the archive names, and it will attempt to apply the data even if it is incorrect.
Using the BACKUP_IN Keyword

You can use the optional BACKUP_IN keyword on the RECOVER FORWARD command to identify the backup data set that you want RUV to recover. The value can be set to one of the following:

- CURRENT_BACKUP (which will select the latest active backup registered in the repository)
- the ddname of the JCL statement that describes the backup data set
- the fully qualified data set name of the backup data set (RUV dynamically allocates the data set)

When the CURRENT_BACKUP value is selected, the recovery utility determines the time range of transactions. Therefore it is not necessary to specify START_TIME or STOP_TIME keyword.

When the CURRENT_BACKUP value is selected, the recovery utility checks for both RUV and SMS backups. If both types of backups exist, the backup with the latest timestamp is used in the recovery. If you do not want to use a particular backup, the BACKUP_METHOD keyword can control which backup methods are used or ignored. See “Using the BACKUP_METHOD Keyword” on page 126 for additional information.

For additional information, see “Restoring Files from Backups” on page 457.

Using the BUFND Keyword

You can use the optional BUFND keyword on the RECOVER FORWARD command to specify the number of data buffers to use. The default is 16, but you can change it to any desired value from 2 to 255. Values beyond 255 will be ignored. VSAM may override invalid combinations of BUFNI/BUFND.

Using the BUFNI Keyword

You can use the optional BUFNI keyword on the RECOVER FORWARD command to specify the number of index buffers to use. The default is 5, but you can change it to any desired value from 2 to 255. Values beyond 255 will be ignored. VSAM may override invalid combinations of BUFNI/BUFND.
Using the INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the RECOVER FORWARD command to show a listing of VSAM files that will be RECOVERED FORWARD, and the name of each RECOVER FORWARD file if it was created by this job. If YES is selected, RECOVER FORWARD processing will not occur but a report will be issued. The INVENTORY_ONLY keyword does not simulate the complete performance of the command, it just reports on what it would have tried to do.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

**YES** Prevent RECOVER FORWARD processing and issue a preliminary report on files that will be archived.

**NO** Allow RECOVER FORWARD processing.

Using the LSR_POOL Keyword

You can use the optional LSR_POOL keyword on the RECOVER FORWARD command to specify whether or not to use LSR POOLS (Local Shared Resource Pools designed to improve performance). The default is YES for KSDS and RRDS data sets. ESDS and IAM data sets do not use LSR POOLS, regardless of how the keyword is set.

**WARNING**
In most cases, setting LSR_POOL to NO will cause severe performance degradation. Therefore, this setting is not recommended unless you use some other product to enhance your VSAM performance.
Using the NEW_NAME Keyword

Use the optional NEW_NAME keyword to recover a VSAM file to a new name. With an alternative VSAM data set name, you can copy production files to test and audit production recovery environments. You may perform many recoveries against the VSAM file without risking damage to an actual production file.

**NOTE**
The NEW_NAME value recovers a data set only to a different VSAM cluster. You must rebuild alternative indexes to recreate a complete data set structure.

**WARNING**
Do not use the NEW_NAME feature with SMS backups that have paths. Use SMS facilities to restore and rename those data sets.

Using the RECOVERY_MODE Keyword

You can use the optional RECOVERY_MODE keyword on the RECOVER FORWARD command to specify the Insertion Strategy for RUV to use when the RECOVER might cause inserts to the VSAM data set that create CI/CA splits. This keyword allows you to control the Insertion Strategy.

The default is NIS (Normal Insertion Strategy), which is good for most files. Files that have a large number of records inserted at one spot might benefit from the RECOVERY_MODE being set to SIS (Sequential Insertion Strategy). Before changing this parameter setting, see the **Warning** on page 475.

Using the SELECTION_EXIT Keyword

You can use the SELECTION_EXIT keyword on the RECOVER FORWARD command to have RUV invoke a user exit routine for each record that it processes during the recovery. It can also be used to override a user exit selection name defined during an ADD or UPDATE VSAM_FILE. See “Using the SELECTION_EXIT User Exit” on page 517. As the value of the SELECTION_EXIT keyword, specify the program name of the exit routine.
Using the START_TIME Keyword

Use the optional START_TIME keyword on the RECOVER FORWARD command to specify the start (earliest) timestamp of the range of journal records to include in the recovery process.

For a RECOVER FORWARD, this timestamp is required if you are using a backup that was not produced by RUV. The timestamp may be required for a point-in-time recovery, depending on the recovery situation. It is not necessary to specify a START_TIME when using the BACKUP_IN(CURRENT_BACKUP) keyword.

Time inputs for RECOVER will accept inputs that go to 5 characters past the decimal of seconds, as in yyyyjjjjhhmmssthtth.

For more information, see “Using the START_TIME Keyword” on page 111.

Using the STOP_TIME Keyword

You can use the optional STOP_TIME keyword on the RECOVER FORWARD command to specify the stop (latest) timestamp of the range of journal records to include in the recovery process. The timestamp may be required when recovering to a specific point in time, depending on the recovery situation. It is not necessary to specify a STOP_TIME when using the BACKUP_IN(CURRENT_BACKUP) keyword.

Time inputs for RECOVER will accept inputs that go to 5 characters past the decimal of seconds, as in yyyyjjjjhhmmssthtth.

For more information, see “Using the RUV ISPF Interface” on page 113.
Using the SUPPRESS_RECOVERY_ERROR Keyword

Use the optional SUPPRESS_RECOVERY_ERROR keyword with the RECOVER FORWARD command to recover to a data set that cannot be set to a correct initial state.

**WARNING**

This feature can lead to invalid recoveries by hiding recovery errors, if SUPPRESS_RECOVERY_ERROR is set to **YES**. You must be careful using this keyword.

You might want to use this keyword in the following scenario:

The data set backups for a particular data set were damaged and cannot be restored, but you still must know what changes were made to the file. You can create an empty data set and try to recover to that specified data set. You will not obtain a good data set with this process, but you will obtain all ADD and UPDATE changes that were made to the data set during the archive process. (You will also obtain the DELETE changes that were applied, but because no records were in this empty data set that you created, you must print the archive and check for DELETE to see each change.)

Although the archive contains the changes, if the recovery runs into an error (for example, the recovery tries to update a record that does not exist), the recovery process usually terminates at this point. The SUPPRESS_RECOVERY_ERROR(YES) keyword instructs RUV to continue the recovery process despite the error.

**NOTE**

In most cases, if the data set was not set to a valid initial state before the recovery, it is not valid after the recovery.

Using the VSAM_FILE Keyword

Use the required VSAM_FILE keyword on the RECOVER FORWARD command to specify the data set to be recovered. Specify the fully qualified data set name that identifies the file as the value of the keyword. You cannot specify masking characters in the value. You can code multiple VSAM_FILE keywords on the command.

If you use any of the optional keywords (such as START_TIME and STOP_TIME), you must code the optional keywords and their values after the DSN of the VSAM_FILE keyword but before the closing parenthesis.
Using RUV for UOW Recovery

To run the RUV unit of work recovery process in batch, execute the RUV utility (program RUVZSM0).

Running an RUV UOW Recovery

To run an RUV UOW recovery, perform the following steps:

1. Determine the VSAM file or files to include for recovery.
2. Build the JCL with the correct space allocation.
3. Build the RECOVER COMPLETED_UNIT_OF_WORK command to specify the files to recover and the options to use. Specify the setting for incremental backups.
4. Run the recovery job or jobs.

Using the RECOVER COMPLETED_UNIT_OF_WORK Command

To perform a UOW recovery for one or more files in batch, use the RECOVER COMPLETED_UNIT_OF_WORK command. All keywords on the command correspond to options that you can control.
Coding the RECOVER COMPLETED_UNIT_OF_WORK Statements

To code a RECOVER COMPLETED_UNIT_OF_WORK statement, use the following syntax:

```
RECOVER COMPLETED_UNIT_OF_WORK
* perform a unit-of-work recovery of one or more VSAM files, one or
* more VSAM groups, or both VSAM files and VSAM groups
  { INVENTORY_ONLY ( NO | YES ) }
  { START_TIME ( yyyyjjhhmmsst | Xhhhhhhhhhhhhhh ) }
  STOP_TIME ( yyyyjjhhmmsst | Xhhhhhhhhhhhhhh )
  VSAM_FILE ( dsn44
    { NEW_NAME ( dsn44 ) }
    { START_TIME ( yyyyjjhhmmsst | Xhhhhhhhhhhhhhh ) }
    { BACKUP_IN ( CURRENT_BACKUP ) | ( ddn8_or_dsn44 ) }
    { SELECTION_EXIT ( program8 ) } }
  { VSAM_FILE ( ... ) }
  VSAM_GROUP ( name32, *, ...)
    { START_TIME ( yyyyjjhhmmsst | Xhhhhhhhhhhhhhh ) }
    { BACKUP_IN ( CURRENT_BACKUP ) }
    { SELECTION_EXIT ( program8 ) }
    { VSAM_NAME_MODEL ( name32 ) }
  )
  { VSAM_GROUP ( ... ) }
  { UNIT_OF_WORK_OUT ( ddn8 ) }
  { UNIT_OF_WORK_IN ( ddn8_or_dsn44 { , ddn8_or_dsn44, ... } ) }
  { ARCHIVE_IN ( ddn8_or_dsn44 { , ddn8_or_dsn44, ... } ) }

;```

Use the RECOVER COMPLETED_UNIT_OF_WORK keyword as shown in the following example to recover two files in a general ledger application. RUV automatically selects the correct archive inputs.

The control statements specify the input backups and VSAM file to use and provide a required stop time of 14:00 PM GMT on September 15, 2009 (Julian Day 2009.258). The STOP_TIME is required for a UOW recovery.

```
RECOVER COMPLETED_UNIT_OF_WORK
  VSAM_FILE(GNL.LEDGER
    STOP_TIME(20092581400000)
    BACKUP_IN(GNL.LEDGER.BKUP)
    INVENTORY_ONLY(NO)
  )
  VSAM_FILE(GNL.PAYROLL
    STOP_TIME(20092581400000)
    BACKUP_IN(GNL.PAYROLL.BKUP)
  )

;```

484 Recovery Utility for VSAM User Guide
Using the ARCHIVE_IN Keyword

Use the optional ARCHIVE_IN keyword to identify an archive data set to use as input to the recovery process.

It is recommended that you allow RUV to select the archive files for use. When you specify the archive files, you override RUV’s normal selection mechanism.

To specify the value of the ARCHIVE_IN keyword, specify one of the following values:

- the ddname of the DD statement that describes the archive data set in the JCL
- the fully qualified data set names of the archive data sets if you want RUV to allocate the data set dynamically

Using the BACKUP_IN Keyword

Use the optional BACKUP_IN keyword to identify the backup data set that you want RUV to recover. If you selected VSAM_GROUP for input, set the BACKUP_IN value to CURRENT_BACKUP to select the latest active backup registered in the repository. If you selected the VSAM_FILE for input, you may specify the fully qualified data set name of the backup data set. When the CURRENT_BACKUP value is selected, the recovery utility determines the time range of transactions. Therefore it is not necessary to specify START_TIME or STOP_TIME keyword.

When the CURRENT_BACKUP value is selected, the recover utility checks for the existence of both RUV and SMS backups. The backup with the latest timestamp will be used in the recovery.
Using the INVENTORY_ONLY Keyword

Use the optional INVENTORY_ONLY keyword to show a listing of VSAM files that will be recovered or the name of each file if it was created by this job. If you specify YES, the recovery processing will not occur but a report will be issued. The INVENTORY_ONLY keyword does not simulate the command, but it reports on the operations that the command would have tried to perform.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

**YES** Prevent the recovery processing and issue a preliminary report on files that will be archived.

**NO** Allow the recovery processing.

Using the REPORT UNIT_OF_WORK Command

Use the REPORT UNIT_OF_WORK(ddn8_or_dsn44, *, ...) command to report information about the completed and incomplete transactions in the UOW archive. See “UNIT_OF WORK Report” on page 423. Use the following syntax:

```
REPORT UNIT_OF_WORK(ddn8)
```

Using the START_TIME Keyword

Use the optional START_TIME keyword to specify the start (earliest) timestamp of the range of journal records to include in the recovery process.

This timestamp is required if you are using a backup that was not produced by RUV. The timestamp could be required for a point-in-time recovery, depending on the recovery situation.

Time inputs for RECOVER will accept inputs that go to 5 characters past the decimal of seconds, as in yyyyjhhmmsssth. For more information, see “Using the START_TIME Keyword” on page 111.
Using the STOP_TIME Keyword

Use the required STOP_TIME keyword to specify the stop (latest) timestamp of the range of journal records to include in the recovery process. The timestamp could be required for a point-in-time recovery, depending on the recovery situation.

Time inputs for RECOVER will accept inputs that go to 5 characters past the decimal of seconds, as in `yyyyjjjhhmmssthtth`.

For more information, see “Using the RUV ISPF Interface” on page 113.

Using the UNIT_OF_WORK_IN Keyword

Use the UNIT_OF_WORK_IN keyword to identify the file that was specified with the UNIT_OF_WORK_OUT keyword in a previous recovery. This file is useful when you are running repetitive forward recoveries. RUV passes the data that it reads from this source to the sort and recover routines without filtering the data by timestamp. RUV still filters the data by VSAM data set name. When you use the UNIT_OF_WORK_IN keyword, you should also specify the START_TIME keyword. The value of the START_TIME keyword should be the earliest STOP_TIME output from the previous recovery.

Using the UNIT_OF_WORK_OUT Keyword

Use the UNIT_OF_WORK_OUT keyword to specify the name of the file where incomplete transactions are written. This file includes information and UOW data records that were not applied as a result of incomplete transactions. It does not include data for transactions that were completed after the stop time specified for the recovery.

To perform repetitive forward recoveries, you can use this output file as input for the next recovery cycle. It is possible to use the same file for input and output for each cycle. The advantage is simplified file management. The disadvantage is no restart point if RUV abends after the ddname has been opened for output. You should consider using a generation data group (GDG) file for input and output if you are running repetitive UOW recoveries.
Using the **VSAM_FILE** Keyword

Use the **VSAM_FILE** keyword to specify the data set to be recovered. Specify the fully qualified data set name that identifies the file as the value of the keyword. You cannot specify masking characters in the value. You can code multiple **VSAM_FILE** keywords on the command.

If you use any of the optional keywords, you must code the optional keywords and their values after the DSN of the **VSAM_FILE** keyword but before the closing parenthesis.

**Using the **VSAM_GROUP** Keyword**

Use the **VSAM_GROUP**(name32) keyword to use a 32-character VSAM group record in the repository. See Chapter 13, “Using VSAM Groups” for further explanation of groups.
Using RUV for Backout Recovery

To perform a backout recovery for a file in batch, use the RECOVER BACKOUT command. Keywords on the command correspond to options you can control.

Performing an RUV Backout Recovery

To run an RUV backout recovery, perform the following steps:

1. Determine the VSAM file or files to include for backout recovery.
2. Build the JCL with the correct space allocation.
3. Build the RECOVER BACKOUT command to specify the files to recover and the options to use.
4. Run the recovery job or jobs.

Using the RECOVER BACKOUT Command

To perform a backout recovery for one or more files in batch, use the RECOVER BACKOUT command. All keywords on the command correspond to options you can control.
Coding the RECOVER BACKOUT Statements

To code a RECOVER BACKOUT statement, use the following syntax:

```
RECOVER BACKOUT
  { INVENTORY_ONLY(NO | YES) }
  VSAM_FILE(dsn44
    { START_TIME(yyyyjjhhmmss) }
    { STOP_TIME(yyyyjjhhmmss) }
    { RECOVERY_MODE(NIS | SIS)}
    { BUFND(nn)}
    { BUFNI(nn)}
    { LSR_POOL(YES | NO) }
    { SELECTION_EXIT(program8) }
  )
  { VSAM_FILE(dsn44 ... ) }
  VSAM_GROUP ( name32, *, ... 
    { START_TIME ( yyyyjjhhmmss | Xhhhhhhhhhhhhhhhh ) }
    { STOP_TIME ( yyyyjjhhmmss | Xhhhhhhhhhhhhhhhh ) }
    { SELECTION_EXIT ( program8 ) }
  )
  { VSAM_GROUP ( ... ) }
  { ARCHIVE_IN(ddn8_or_dsn44, ...) }
; 
```

The following example shows a backout recovery of two files in a general ledger application. RUV can invoke a user exit routine during processing of the files.

```
RECOVER BACKOUT
  VSAM_FILE(GNL.LEDGER
    START_TIME(20092540000000)
    SELECTION_EXIT(GNLEXIT)
  )
  VSAM_FILE(GNL.PAYROLL
    START_TIME(20092540000000)
    SELECTION_EXIT(GNLEXIT)
  )
; 
```
Using the ARCHIVE_IN Keyword

You can use the optional ARCHIVE_IN keyword on the RECOVER BACKOUT command to identify an archive data set to use as input to the recovery process. It is recommended that you let RUV select the archive files for use. When you specify the archive files, you override RUV’s normal selection mechanism. As the value of the ARCHIVE_IN keyword, specify the ddname of the DD statement that describes the archive data set in the JCL, or specify the fully qualified data set names of the archive data sets to have RUV allocate the data set dynamically.

WARNING

The LSR_POOL, BUFNI, BUFND, and RECOVERY_MODE parameters can greatly affect the performance of a RECOVER. Before you change from the defaults, it is strongly recommended that you test the effects of changing these parameters before implementing them in production environments.

Using the BUFND Keyword

You can use the optional BUFND keyword on the RECOVER BACKOUT command to specify the number of data buffers to use. The default is 16, but you can change it to any desired value from 2 to 255. Values beyond 255 will be ignored. VSAM may override invalid combinations of BUFNI/BUFND.

Using the BUFNI Keyword

You can use the optional BUFNI keyword on the RECOVER BACKOUT command to specify the number of data buffers to use. The default is 5, but you can change it to any desired value from 2 to 255. Values beyond 255 will be ignored. VSAM may override invalid combinations of BUFNI/BUFND.
Using the INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the RECOVER BACKOUT command to prevent RECOVER BACKOUT processing and show a listing of VSAM files that will be included in RECOVER BACKOUT, and the name of each RECOVER BACKOUT file if it was created by this job. The INVENTORY_ONLY keyword does not simulate the complete performance of the command, but reports on what it would have tried to do.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

**YES** Prevent RECOVER BACKOUT processing and issue a preliminary report on files that will be archived.

**NO** Allow RECOVER BACKOUT processing.

Using the LSR_POOL Keyword

You can use the optional LSR_POOL keyword on the RECOVER BACKOUT command to specify whether or not to use LSR POOLS (Local Shared Resource Pools designed to improve performance). The default is YES for KSDS and RRDS data sets. ESDS and IAM data sets do not use LSR POOLS, regardless of how the keyword is set.

---

**WARNING**

In most cases, setting LSR_POOL to NO will cause severe performance degradation.

Therefore, this setting is not recommended unless you use some other product to enhance your VSAM performance.

---

Using the RECOVERY_MODE Keyword

You can use the optional RECOVERY_MODE keyword on the RECOVER BACKOUT command to specify the Insertion Strategy for RUV to use when the RECOVER might cause inserts to the VSAM data set that create CI/CA splits. This keyword allows you to control the Insertion Strategy.

The default is NIS (Normal Insertion Strategy), which is good for most files. Files that have a large number of records inserted at one spot might benefit from the RECOVERY_MODE being set to SIS (Sequential Insertion Strategy). Before changing this parameter setting, see the Warning on page 491.
Using the SELECTION_EXIT Keyword

You can use the SELECTION_EXIT keyword on the RECOVER BACKOUT command to have RUV invoke a user exit routine for each record that it processes during the recovery. It can also be used to override a user exit selection name defined during an ADD or UPDATE VSAM_FILE. See “Using the SELECTION_EXIT User Exit” on page 517. As the value of the SELECTION_EXIT keyword, specify the program name of the exit routine.

Using the START_TIME Keyword

Use the required START_TIME keyword on the RECOVER BACKOUT command to specify the start (earliest) timestamp of the range of journal records to include in the recovery process.

Time inputs for RECOVER will accept inputs that go to 5 characters past the decimal of seconds, as in \texttt{yyyyjjjjhhmmssstthth}.

For more information, see “Using the START_TIME Keyword” on page 111.

Using the STOP_TIME Keyword

You can use the optional STOP_TIME keyword on the RECOVER BACKOUT command to specify the stop (latest) timestamp of the range of journal records to include in the recovery process. The timestamp may be required for recovery to a specific point in time depending on the recovery situation.

Time inputs for RECOVER will accept inputs that go to 5 characters past the decimal of seconds, as in \texttt{yyyyjjjjhhmmssstthth}.

For more information, see “Using the RUV ISPF Interface” on page 113.
Using the VSAM_FILE Keyword

Use the required VSAM_FILE keyword on the RECOVER BACKOUT command to specify the data set to be recovered. As the value of the keyword, specify the fully qualified data set name that identifies the file. You cannot specify masking characters in the value. You can code multiple VSAM_FILE keywords on the command.

If you use any of the optional keywords (such as START_TIME and STOP_TIME), you must code the optional keywords and their values after the DSN of the VSAM_FILE keyword but before the closing parenthesis.
Using the ISPF Interface for Recovery

The RUV ISPF interface can generate and submit recovery JCL.

For general information about the RUV ISPF interface, see “Using the RUV ISPF Interface” on page 113.

Accessing the VSAM Sphere Recovery Panels

To access the VSAM sphere recovery panels in the RUV ISPF interface, perform the following steps:

1 Start the ISPF interface as described in “Starting the RUV ISPF Interface” on page 113.

   The RUV Options panel (Figure 19 on page 115) is displayed. The Subsystem ID field identifies the active subsystem that is currently selected.

2 If you want to use a different subsystem, change any character in the Subsystem ID field, and press Enter.

   The Subsystem List pop-up window (Figure 18 on page 114) is displayed with the names of active subsystems. Select the subsystem that you want to use, and press Enter to return to the RUV Options panel.

3 Type 1 (Perform a VSAM Recovery) in the choice entry field, and press Enter.

   The VSAM Sphere Recovery - Overview panel (not shown) is displayed. This panel provides basic introductory information about the VSAM sphere recovery panels.

4 Press Enter.

   The VSAM Sphere Recovery - Control Panel (Figure 132 on page 496) is displayed.
Using the VSAM Sphere Recovery - Control Panel

The VSAM Sphere Recovery - Control Panel (Figure 132) leads you through the guided recovery process. This panel and subsequent panels present typical decisions that you might need to make for a successful recovery. Based on your responses, RUV can generate JCL that you can submit or save for modification and use at another time.

Figure 132  VSAM Sphere Recovery Control Panel

By default, the two steps that must be performed for a forward recovery using the current backup files (Select VSAM files and Generate JCL) are preselected with a forward slash on this panel. To perform this type of recovery, press Enter and answer the questions as they are presented.

After you complete each recovery preparation step, RUV returns you to this control panel. The informational messages for each option will change after each step is completed.

You can select the guided recovery steps in any sequence and modify recovery options. After making changes in the options, you must regenerate the JCL before submitting the recovery job. Novice users should use the default values presented in the guided recovery process.
On the following pages, you will find examples of the guided recovery panels displayed at each step in the guided recovery process. The options for customizing the recovery process are described with each panel.

<table>
<thead>
<tr>
<th>Guided Recovery Option</th>
<th>Go To Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select VSAM files for recovery</td>
<td>498</td>
</tr>
<tr>
<td>Select time range</td>
<td>501</td>
</tr>
<tr>
<td>Select backup files</td>
<td>503</td>
</tr>
<tr>
<td>Select forward or backout recovery</td>
<td>505</td>
</tr>
<tr>
<td>Select new names for VSAM files</td>
<td>506</td>
</tr>
<tr>
<td>Generate JCL for recovery</td>
<td>508</td>
</tr>
<tr>
<td>Save recovery JCL</td>
<td>510</td>
</tr>
<tr>
<td>Submit JCL for execution</td>
<td>511</td>
</tr>
</tbody>
</table>
Using the VSAM Sphere Recovery - Filters Panel

When you select the Select VSAM files for recovery option on the VSAM Sphere Recovery - Control Panel, the VSAM Sphere Recovery - Filters panel (Figure 133) is displayed. This panel allows you to filter the list of VSAM files that are presented for selection. You can specify filters for VSAM groups, VSAM files, or both by using a fully qualified data set name or wildcard masking characters.

**Figure 133  VSAM Sphere Recovery Filters Panel**

If you want to apply a VSAM group filter, type the mask in the VSAM Group Name filter mask field and type any character to the left of the Display VSAM Group Names field. When you press Enter, the VSAM Sphere Recovery - Group Names panel (Figure 134 on page 499) is displayed.

If you want to apply a VSAM file filter, type the mask in the VSAM File Name filter mask field and type any character to the left of the Display VSAM File Names field. When you press Enter, the VSAM Sphere Recovery - VSAM File Names panel (Figure 135 on page 500) is displayed.

If you want to have all VSAM file names that match the VSAM file name filter marked as Selected on the next panel, type any character to the left of the Preselect all VSAM files field.
Using the VSAM Sphere Recovery - Group Names Panel

When you enter a mask in the VSAM Group Name filter mask field and select the Display VSAM Group Names field on the VSAM Sphere Recovery - Filters panel (Figure 133 on page 498), the VSAM Sphere Recovery - Group Names panel (Figure 134) is displayed. This panel lists the VSAM group or groups that match the specified mask.

**Figure 134  VSAM Sphere Recovery - Group Names Panel**

Select one or more of the VSAM group names, and press Enter. The VSAM Sphere Recovery - VSAM File Names panel (Figure 135 on page 500) is displayed. All files in the group are marked as selected.
Using the VSAM Sphere Recovery - VSAM File Names Panel

When you enter a mask in the VSAM File Name filter mask field and select the Display VSAM File Names field on the VSAM Sphere Recovery - Filters panel (Figure 133 on page 498), or when you select one or more of the VSAM group names on the VSAM Sphere Recovery - Group Names panel (Figure 134 on page 499), the VSAM Sphere Recovery - VSAM File Names panel (Figure 135) is displayed. This panel lists the VSAM files that match the specified mask or that are members of the selected group or groups.

The Status field indicates whether an individual files is selected for recovery. All files in the list are marked as SELECTED if you selected the Preselect all VSAM files field on the VSAM Sphere Recovery - Filters panel. All files that are members of a selected group are also marked as SELECTED. Files that have a status of NOT SELECTED are not included in the recovery.

Figure 135  VSAM Sphere Recovery - VSAM File Names Panel

To change the status of a file, type any character to the left of the file name and press Enter. Multiple files can be selected and changed at the same time.

When the files have the correct statuses, press Enter to accept the statuses and return to the VSAM Sphere Recovery - Control Panel (Figure 132 on page 496). The comments to the right of the Select VSAM files for recovery field are changed to VSAM file(s) selected. Press Enter to continue the guided recovery process with the next option that you have selected.
Specify a Recovery Time Range Panel

To display the VSAM Sphere Recovery - Time Range panel (Figure 136), select the Select Time Range option on the VSAM Sphere Recovery - Control Panel. For RUV, the default type of recovery is a forward recovery and the desired stop time is assumed to be the current time. The desired start time is determined by examining all appropriate and registered backups and archives.

For forward recovery, specifying the Start Time is helpful in limiting the data to be considered in the recovery process.

For a backout recovery, a Start Time is required but the Stop Time is optional. Enter the start (earliest) time for the range of journal records to be include in the recovery process. For more information, see “Manual Forward Recovery” on page 466.

Figure 136  VSAM Sphere Recovery - Time Range Panel

To enter a time, you may

- Select a time field and press Enter to use the RUV default values.
- Enter the time using the yyyydddhhmmss timestamp.
- Enter the time in hexadecimal format.
- Omit the timestamp to use the RUV-generated or default value.

RUV shows a value of 2042.260 when it does not have an end date and zeros when it does not have a beginning date.
For a UOW recovery, you must supply the proper Start Time. The Start Time for this recovery is the same as the Stop Time printed out by the previous recovery. To maintain data integrity enter the use store clock format to enter from 1 to 16 hex digits.

For a UOW recovery on a system with multiple CICS, at the end of the recovery RUV prints out the Stop Time for each APPLID in the recovery. The last Stop Time in that recovery should be used as the first Start Time for the next recovery.
Select Backup Files Panel

To display the VSAM Sphere Recovery - Select Backup panel (Figure 137), select the Select backup files option on the VSAM Sphere Recovery - Control Panel.

This panel lists the VSAM files to be recovered that have multiple backups. You will be allowed to specify which backup you want to use.

**Figure 137  VSAM Sphere Recovery Select Backup Panel**

To select a backup, type a selection character (/ or S) in the field to the left of the VSAM File Name line, and press Enter. The Backup List panel is displayed (Figure 138).

The name of the VSAM file for which you are selecting the VSAM DSN backup is displayed above the list of the Backup DSN file names.
To select a specific backup, type a selection character (/ or S) in the field to the left of the Backup DSN name and press Enter. Only one backup for a VSAM DSN can be selected. RUV issues an error if multiple backups are selected.

**NOTE**

When you select a backup file, RUV specifies the type of recovery as a forward recovery. The backout recovery option is not available.

If there are multiple backups for another VSAM file, the VSAM Sphere Recovery - Select Backup panel (Figure 137) is displayed again with the related archive and backup names.
Select Forward or Backout Recovery Panel

To specify the type of recovery to perform, select the Select Forward or Backward recovery option on the VSAM Sphere Recovery Control Panel. A Recovery Type pop-up panel (Figure 139) is displayed.

**Figure 139 VSAM Sphere Recovery Type Panel**

As shown on the Control Panel, a Forward Recovery the default recovery type. To select the type of recovery, type the corresponding numeric value in the selection field and press Enter.

**NOTE**

If you have previously selected a backup file from the Control Panel, RUV assumes it is to perform a forward recovery and that decision can not be changed on this Recovery Type pop-up panel.
Select New Names Panel

To display the VSAM Sphere Recovery - New Names panel (Figure 140), select the Select new names for VSAM files option on the VSAM Sphere Recovery - Control Panel. A new name can be specified for the forward recovered VSAM files but not for backout recovered files.

Figure 140  VSAM Sphere Recovery New Names Panel

There are five different options for modifying the DSN names; however, only one rename option can be used for the files in a specific recovery job.

In the example in Figure 140, the option is selected with a slash character, and the desired prefix (RUV.) is typed at the end of the line. All recovered files in this job will have the same prefix.

To change the DSN file name, select the desired new name option, and press Enter. A pop-up panel (Figure 141) is displayed. You can review, and edit if needed, the new name of all the selected files.
Use the following keys to respond to each new name on the pop-up panel:

**Enter**
Accept the file name as displayed

**F12** Discard the changes made on the pop-up panel, and use the new name that was displayed when the pop-up opened

**F10** Stop reviewing the new names, discard any changes that had been made, and return to the VSAM Sphere Recovery Control Menu.
To display the Generated JCL panel (Figure 142), select the **Generate JCL for Recovery** option on the VSAM Sphere Recovery Control Panel.

RUV collects and analyzes the information needed to generate a recovery job stream for the selected VSAM files. If all required recovery resources are available, RUV generates the JCL code and displays it for your review.

**Figure 142  SAM Sphere Recovery Panel - Generate JCL (ISPF Edit)**

The JCL is displayed in an ISPF edit window. Use the exit key (F3) to exit the ISPF editor window. This JCL is held in memory as a temporary file. It will be deleted when you exit this RUV session unless you save it.

You can issue a Save command or a Submit command from this panel, or you can use the other guided recovery options on the VSAM Sphere Recovery Control Panel.

If any of the required recovery resources are not available, RUV displays a detailed error message (Figure 143). You can review the messages, correct any errors, and repeat the process to generate the corrected JCL.
Figure 143  VSAM Sphere Recovery Panel - Problem Warning

These VSAM files may not be recoverable. Select the VSAM DSN to see details and potential solutions.

Command ===> ______________________________________________________________
F1=Help      F3=Exit     F10=Actions  F12=Cancel
Save Recovery JCL Panel

To save the JCL, select the **Save Recovery JCL** option on the VSAM Sphere Recovery - Control Panel. A pop-up Save JCL As panel is displayed.

If you made any changes in the recovery options after generating the JCL, you must regenerate the JCL before saving the JCL.

**Figure 144  VSAM Sphere Recovery - Control Panel - Save JCL**

In the Save JCL As pop-up window, type a valid DSN, and press Enter.

If the DSN is invalid or you do not have permissions to save the JCL, an error message is displayed. Correct the problem and repeat the Save Recovery JCL procedure.

Use the following keys to respond to JCL DSN name on the pop-up panel:

- **F10**  Accept the file name as displayed
- **F12**  Discard the changes made on the pop-up panel, and return to the VSAM Sphere Recovery Control Menu.
Submit JCL Panel

To submit a JCL, select the Submit JCL for Execution option on the VSAM Sphere Recovery Control Panel.

The JCL is submitted as a background process. No panels or pop-up windows are displayed.

To verify that the JCL was submitted, RUV displays an informational message:

- The message on the Submit JCL for execution line changes from <Optional> to <JCL Submitted>
- A message (containing the job name, job number, and status), appears below the Make your selection line.

Figure 145  VSAM Sphere Recovery - Control Panel - Submit JCL
## Understanding RUV Exception Reasons

Table 19 lists the exception reasons that RUV uses when it processes journal records during recovery.

### Table 19 Journal Record REJECT Exception Reasons (Part 1 of 2)

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rcd is accumed out</td>
<td>For a forward recovery, a journal record was eliminated during an ACCUM operation when a more recent update activity journal record was found. For a backward recovery, a journal record reflecting earlier activity was found.</td>
</tr>
</tbody>
</table>
| Add followed update  | Journal records are marked as “Add followed update” during an ACCUM operation when one of the following occurs:  
   - an update journal record is followed chronologically by an add record for the same key  
   - an add journal record is followed immediately by another add record for the same key |
| Keylen not match mstr| A journal record is marked “Keylen not match mstr” when one of the following conditions is true:  
   - The key length of the VSAM KSDS being recovered has changed since the journal record was created.  
   - The journal record requires that a delete operation be performed on a VSAM ESDS file.  
   - A VSAM ESDS or RRDS is being recovered and the key length contained in the journal record is not four.  
   - The journal record for the VSAM RRDS contains an invalid relative record number. |
| Master illogical     | A journal record is marked “Master illogical” when one of the following conditions is true:  
   - An Update function BACKOUT request is being processed and the journal record represents an add to an ESDS.  
   - The journal record is for an add operation but a master file record with the same key exists.  
   - The journal record is for an Update function request but a master file record with the corresponding key does not exist. |
Rcd not required
A journal record is marked “Rcd not required” when one of the following conditions is true:
- It is for a forward recovery delete operation and the corresponding master file record does not exist.
- It is for an add operation during an Update function BACKOUT and the corresponding master file record does not exist.

Rcd on file already
A journal record is marked “Rcd on file already” when the Recovery utility compares it to the master file record image and finds that they are identical.
Invoking IDCAMS from an RUV Job Step

You can perform the IDCAMS delete/define operations in separate steps from the RUV restore or recovery process, or you can have RUV invoke IDCAMS to perform the delete/define operations in the same job step with the restore or recovery process.

RUV honors the IDCAMS return code. RUV does not display the SYSPRINT output from IDCAMS if the return code from IDCAMS is zero. If the return code is greater than zero or you have selected the detail or full level of reporting, RUV includes all IDCAMS SYSPRINT output in the RUV job output.

Using the IDCAMS Commands

To mark the beginning of a series of IDCAMS commands that you want RUV to pass to IDCAMS during the RUV job step, use the ENTER_IDCAMS command, as shown in the following example. RUV passes all statements from the semicolon after the ENTER_IDCAMS command to the last character before the EXIT_IDCAMS command.

Coding the ENTER_IDCAMS and EXIT_IDCAMS Commands

To code an UPDATE VSAM_FILE statement, use the following syntax:

```
ENTER_IDCAMS;
  [ statements to pass to IDCAMS ]
EXIT_IDCAMS;
```

Use the ENTER_IDCAMS and EXIT_IDCAMS commands as shown in the following example:

```
* define of ESDS test file
ENTER_IDCAMS;
  DELETE (RUVRUV.TEST.EBASE) CLUSTER
  SET MAXCC=0
  DEFINE CL (NAME(RUVRUV.TEST.EBASE) -
    NONINDEXED -
    RECSZ(512 8192) -
    TRK (8 4) -
    VOLUME(DEV233) -
    SHAREOPTIONS(2,3) -
  )
EXIT_IDCAMS;
```
Introducing a Delay into an RUV Process

During recovery or other automated RUV processes, you might need to introduce a delay between the time when the process is submitted for execution and the time when the process starts executing.

For example, an application job step has abended and caused the need for a backout recovery of a VSAM file. RUV automatically detects the need for a backout recovery and submits the JCL to recover the VSAM file. However, the abended job step has not yet released the enqueue on the VSAM file, and therefore the RUV backout recovery job step fails because it cannot obtain an enqueue on the VSAM file.

In this case, you could use the TEST DELAY command of the RUVZSM0 utility to give the abended job step more time to complete its cleanup processing before the backout recovery job step attempts to obtain an enqueue.

Coding the TEST DELAY Command

Code the TEST DELAY command as shown in the following syntax:

```
TEST DELAY(ss)
```

The value of the DELAY keyword is the length of time, in seconds, for RUV to pause before continuing with execution.

Using the TEST DELAY Command

The following example shows the use of the TEST DELAY command in a RECOVER BACKOUT job step to request a 15-second pause after the job step is submitted and before it starts execution:

```
SET SUBSYSTEM(RUV$);
TEST DELAY(15);
RECOVER BACKOUT
  VSAM_FILE(dsn44)
  ARCHIVE_IN(dsn44)
...`
```
You can also include the TEST DELAY command in a JOB_JCL model, as shown in the following example:

```
UPDATE JOB_JCL(name32)
   //<JOBNAME...
   ...
   //SYSIN DD *
   TEST_DELAY(25)
   <RUV_CONTROL>
   ...
```
Using the SELECTION_EXIT User Exit

RUU provides a user exit for the following commands:

- RECOVER BACKOUT
- RECOVER FORWARD
- PRINT ARCHIVE_IN (useful for getting a print of the journal)

The user exit is active during processing when you specify an exit program value for the SELECTION_EXIT(program) keyword; you must also provide the RPTOUT DD statement in the RUUVZSM0 JCL.

The user exit is called for each record to be processed. For example, this exit can be useful to an auditing program that may want to examine whether a particular RUV-recorded VSAM record has been read, added, updated, or deleted.

RUU loads the link-edited user exit program you specify with the SELECTION_EXIT(program) keyword. It then uses standard operating system linkage conventions to call the program.

--- WARNING ---

The parameter list passed to the user program will contain an address above the 16 MB line. This condition requires that you link the user exit program with AMODE(31) (access mode 31). The RMODE(24, 31, or ANY) (residency mode) may be whatever the user program requires.

A sample user exit program is found in the ZZUZSEL member of the RUVCNTL library.

User Exit Calls

The user exit is called as follows:

- at beginning of processing
- once for each record
- at the end of the processing

The first invocation of the user exit is before any record is processed. Parameter 07 points to a one-byte action code x'00', which indicates that this call is the initial call.
The exit is invoked once for each record to be processed and provides the full set of parameters with their corresponding pointer or value. Only the archive records that represent user recovery data are passed to the exit; internal control records are not passed. The user-token parameter is the same value that is established by the user program during the initial call.

The last invocation to the user exit is at the end of the data. If the user has used the “user token” value as a place holder for a persistent storage address, the user exit should release this storage before returning from this last call. Parameter 07 points to a one-byte action code x'FF', which indicates that this call is the last call.

### User Exit Action Codes

Parameter 07 points to a 2-byte action code. The first byte identifies the type of action, and the second byte identifies the source of the call.

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Action and Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'00'</td>
<td>X'00'</td>
<td>The exit is initializing.</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>The exit is processing a “data add” record, which indicates an intent to add a data record. The exit was called during processing of the RUV RECOVER command. CICS journals both the intent to add a record and the successful completion of the add operation. The operation can fail for various reasons (such as a duplicate base key, a duplicate path key, a data set out-of-space condition, and so on).</td>
</tr>
<tr>
<td>A</td>
<td>P</td>
<td>The exit is processing a “data add” record, which indicates an intent to add a data record. The exit was called during processing of the PRINT ARCHIVE_IN command.</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>The exit is processing a “data add complete” record, which indicates that a data record was added successfully. The exit was called during processing of the RUV RECOVER command.</td>
</tr>
</tbody>
</table>
### User Exit Action Codes

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Action and Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>P</td>
<td>The exit is processing a “data add complete” record, which indicates that a data record was added successfully. The exit was called during processing of the PRINT ARCHIVE_IN command.</td>
</tr>
<tr>
<td>D</td>
<td>P</td>
<td>The exit is processing a “delete” record, which indicates that a data record was erased without being read first. The exit was called during processing of the PRINT ARCHIVE_IN command.</td>
</tr>
<tr>
<td>L</td>
<td>P</td>
<td>The exit is processing a “logical unit of work” record. The exit was called during processing of the PRINT ARCHIVE_IN command. At the successful completion of all tasks that are required for a correct recovery to a point in time, a logical unit of work boundary has been reached. Records are applied or not applied during recovery depending on whether they belong to the logical unit of work.</td>
</tr>
<tr>
<td>R</td>
<td>A</td>
<td>The exit is processing a “data read for update” record. The exit was called during processing of the RUV RECOVER command.</td>
</tr>
<tr>
<td>R</td>
<td>P</td>
<td>The exit is processing a “data read for update” record. The exit was called during processing of the PRINT ARCHIVE_IN command.</td>
</tr>
<tr>
<td>U</td>
<td>A</td>
<td>The exit is processing a “data update delete” record, which indicates that a data record was read for update and then deleted. The exit was called during processing of the RUV RECOVER command.</td>
</tr>
<tr>
<td>U</td>
<td>P</td>
<td>The exit is processing a “data update delete” record, which indicates that the record was read for update and then deleted. The exit was called during processing of the PRINT ARCHIVE_IN command.</td>
</tr>
<tr>
<td>X’ff</td>
<td>X’00’</td>
<td>The exit is terminating because it has reached the end of data.</td>
</tr>
</tbody>
</table>
User Exit Return Codes

The user exit provides one of the following return codes:

00  continue to call the user exit
04  do not print
08  do not call the user exit again
16  terminate all processing

If the user exit specifies either 08 or 16 as the return code, no call is made to the user program at the end of data process.

User Exit Parameter List

Table 20 describes the structure of the parameter list that is passed to the user exit:

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>parm_length</td>
<td>4</td>
<td>the parameter list length (includes itself)</td>
</tr>
<tr>
<td>02</td>
<td>dsn</td>
<td>4</td>
<td>a pointer to the 44-character data set name to which this record is associated</td>
</tr>
<tr>
<td>03</td>
<td>key length</td>
<td>4</td>
<td>the key length that is associated with the data set name</td>
</tr>
<tr>
<td>04</td>
<td>key</td>
<td>4</td>
<td>a pointer to the key for this record</td>
</tr>
<tr>
<td>05</td>
<td>data length</td>
<td>4</td>
<td>the length of the data record</td>
</tr>
<tr>
<td>06</td>
<td>data record</td>
<td>4</td>
<td>a pointer to the data record</td>
</tr>
<tr>
<td>07</td>
<td>action</td>
<td>4</td>
<td>a pointer to a 2-byte action value</td>
</tr>
<tr>
<td>08</td>
<td>jobname</td>
<td>4</td>
<td>a pointer to an 8-byte field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a batch job caused this record, it is an 8-byte jobname. If CICS caused this record, the field is an applid.</td>
</tr>
<tr>
<td>09</td>
<td>timestamp</td>
<td>4</td>
<td>a pointer to an area of three or six contiguous fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Starting UOW 8 byte TOD store clock value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Starting UOW 4 byte packed date ‘0CYYJJJs’</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Starting UOW 4 byte packed time ‘HHMMSSTs’</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ending UOW 8 byte TOD store clock value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ending UOW 4 byte packed date ‘0CYYJJJs’</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ending UOW 4 byte packed time ‘HHMMSSTs’</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Starting or ending UOW fields may be all x’00’ to indicate an incomplete unit of work</td>
</tr>
</tbody>
</table>
Table 20  Parameter List Structure (Part 2 of 2)

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>origination</td>
<td>4</td>
<td>a pointer to a 1 byte value that indicates the origin:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x'01' — Batch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following values are for CICS only:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x'02' — 0.3.3.x RPCV version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x'03' — 4.2.x.4 RPCV version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x'04' — 5.0.0.0 RPCV version or later</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x'05' — 0.0.0.1 Log Block version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x'06' — 0.0.0.1 CICS journal input</td>
</tr>
<tr>
<td>11</td>
<td>user token</td>
<td>4</td>
<td>a user-defined token</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>On the first call to the user exit, this value will be x'00000000'. You may set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>this value only on the first call. The value you set will be returned on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>subsequent calls.</td>
</tr>
<tr>
<td>12</td>
<td>task number</td>
<td>4</td>
<td>CICS only - a pointer to a 4-byte packed decimal value</td>
</tr>
<tr>
<td>13</td>
<td>TRANID</td>
<td>4</td>
<td>CICS only - a pointer to a 4-byte character value known as the CICS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>transaction ID</td>
</tr>
<tr>
<td>14</td>
<td>TERMID</td>
<td>4</td>
<td>CICS only - a pointer to a 4-byte character value that is the terminal ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>as defined by CICS</td>
</tr>
</tbody>
</table>
This chapter describes how to use the Recovery Utility for VSAM (RUV) product in a disaster recovery plan. This chapter includes the following topics:

Overview ................................................................. 524
   An Overview of Disaster Recovery ............................. 524
   Scope and Situation ............................................... 524
   General Tools for Successful Disaster Recovery ........... 525
   Tools for Disaster Recovery from BMC Software .......... 525
   Summary of Disaster Recovery Processes ................. 525
Phase 1—Preparing for Disaster Recovery .................. 526
   Establishing RUV Groups ....................................... 526
   Obtaining Backups of Assets ................................. 527
   Verifying the Recovery Process .............................. 528
Phase 2—Performing Disaster Recovery .................... 529
   Restoring the RUV Repository and Starting RUV ....... 529
   Understanding the Potential for Lost Data After Recovery 530
Phase 3—Evaluating the Disaster Recovery ............... 531
Overview

This section provides a brief overview of disaster recovery, discusses some considerations to keep in mind as you read this chapter, and summarizes the process of managing a disaster recovery with the RUV product.

The focus of the information in this chapter is on the disaster recovery of RUV and VSAM data sets. Disaster recovery in the overall OS/390 environment is mentioned briefly only as an aid to understanding how RUV fits into an overall plan. This chapter assumes that your disaster recovery site is already prepared for execution of necessary system software and utilities (including RUV).

An Overview of Disaster Recovery

Disaster recovery consists of the preparations and procedures for resuming critical data processing tasks at a remote site if a disaster (such as a fire, flood, or storm) causes a lengthy outage at the home site. The goal is to have all necessary VSAM files available and all necessary applications running again in the least amount of elapsed time (given the resources at the disaster recovery site), with the least loss of data and with complete data integrity.

Scope and Situation

Disaster recovery is different in scope from application recovery. With disaster recovery, the amount of data to recover is massive because most or all of the databases at your site might need to be recovered. The disaster recovery site might have subtle differences from the home site, differences that may cause problems with the recovery.

The business of the entire company may be halted while disaster recovery is performed. Disaster recovery drills are often highly visible in the corporate environment, adding the stress of executive observation of your activities. And while no actual recovery is a relaxing, enjoyable event, a real disaster recovery may be made even more stressful by concerns about the safety of lives and property that remain at the home site.
General Tools for Successful Disaster Recovery

Successful disaster recovery involves careful planning, knowledge of your processing environment, commitment of time and resources, and a set of tools and utilities. OS/390 and RUV provide basic functions and utilities for tasks such as taking backups, logging changes, allocating data sets, and performing VSAM recoveries.

Tools for Disaster Recovery from BMC Software

Enhanced tools for ensuring the success of your disaster recovery effort are provided by BMC Software. These tools provide for faster backups and recoveries, with maximum use of available resources and maximum preservation of data integrity. These tools also provide functions that are not available in other products.

The Recovery Utility for VSAM (RUV) functions and utilities are ideal for helping you in the disaster recovery effort. You can use RUV reports to determine which backups are available and the location and VOLSER of the backups.

You can use the NEW_NAME feature of restore and recover to practice disaster recovery at the home site.

Summary of Disaster Recovery Processes

This chapter presents procedures for disaster recovery with RUV in three phases:

“Phase 1—Preparing for Disaster Recovery” on page 526 describes how to get ready for disaster recovery with RUV.

“Phase 2—Performing Disaster Recovery” on page 529 describes tasks to perform in a disaster recovery simulation or actual disaster recovery situation.

“Phase 3—Evaluating the Disaster Recovery” on page 531 describes some changes you may want to make to your disaster recovery plan after testing it.
Phase 1—Preparing for Disaster Recovery

This section describes the first—and most complex and important—phase of disaster recovery: preparation. The preparation phase takes most of the time and effort in the overall disaster recovery effort. With proper preparation, a real or simulated disaster recovery is a simple matter of performing the procedures to use the elements you have already created and verified. Without proper preparation, no disaster recovery scenario can be successful.

The preparation phase consists of the following procedures:

1. Establishing RUV groups
2. Obtaining backup of assets
3. Verifying the recovery process

Establishing RUV Groups

You initiate RUV functions against a single VSAM file or groups of VSAM files. Because a disaster recovery situation typically is different from an application recovery situation, you might want to create one or more groups especially with disaster recovery in mind. Here are some approaches:

- You could use the group definitions already defined at the local site. If personnel are already familiar with the backup and restore groups, training and set up time is saved and coordination efforts between the local and disaster sites are reduced.

- You could create a group for each application and recover the groups in order of application priority. For example, your organization may need to get the order processing application running right away, while the payroll application can wait until the end of the pay period. This approach provides flexibility and allows you to change priorities easily for each group.

To establish RUV groups for disaster recovery, perform the following steps:

1. Evaluate possible groups, and select one that is appropriate for your needs.

2. Validate disaster recovery groups regularly.

To ensure that the groups that you have created contain all objects that need to be recovered, set up regularly scheduled jobs to restore and recover the groups. Use the NEW_NAME feature of RUV to test the recovery locally.
Obtaining Backups of Assets

RUV is capable of creating multiple backup copies with a single pass of the data. One copy remains available locally, and the other is shipped to an off-site vault. The local copy should be registered as active and have a location code that identifies the local site. The vault copy should be registered as inactive and have a location code that identifies the vault.

You must obtain backups of all assets that are needed for disaster recovery and send the backups to the disaster recovery site. Perform the following steps:

1. Back up the VSAM data sets, as shown in the following example:

   ```
   BACKUP VSAM_FILE(GNLP.ACCOUNTS.MASTER)
   BACKUP_OUT(DDI,
     COMMENT("Local site")
     STATUS(ACTIVE)
     LOCATION(LOCAL)
   )
   BACKUP_OUT(DD2,
     COMMENT("Vault")
     STATUS(INACTIVE)
     LOCATION(VAULT)
   );
   ```

2. Back up the RUV repository in the same fashion, as shown in the following example:

   ```
   BACKUP REPOSITORY
   BACKUP_OUT(RPSTRY1,
     COMMENT("Local site")
     STATUS(ACTIVE)
     LOCATION(LOCAL)
   )
   BACKUP_OUT(RPSTRY2,
     COMMENT("Vault")
     STATUS(INACTIVE)
     LOCATION(VAULT)
   );
   ```

3. Obtain backups of all RUV-related data sets, including the product load libraries, IDCAMS DELETE / DEFINE commands for the RUV repository and other VSAM data sets, SMS-based backups, and the SMS catalog.

4. Send all backups and associated reports to the disaster recovery site. Be sure to make shipments to the disaster recovery site frequently enough to meet your service-level agreement for disaster recovery.
Verifying the Recovery Process

The final procedure in successfully preparing for disaster recovery is verifying the recovery process. You can perform this procedure at the local site. You might want to check with your system security administrator and create a special system authorization facility (SAF ID) to prevent access to the production VSAM files and subsystem.

Perform the following steps to verify the recovery process:

1. Use the RUV reports and related reports from other products to verify that all pertinent files have been backed up and are available.

2. Start the test RUV subsystem and restore the RUV repository backup. Take note of the restore time, as large repositories may affect the time requirements for recovery.

3. Alter the backup and archive statuses as appropriate. The following commands assume all off-site copies have been identified with a LOCATION(VAULT) parameter. The commands make the local site files unavailable and change files at the VAULT location to an active status.

   Change all backups and archives to an inactive status with the following commands:

   ```
   UPDATE ARCHIVE_FILE(*)
   FILTER_BY(LOCATION(*))
   STATUS(INACTIVE);
   UPDATE BACKUP_FILE(*)
   FILTER_BY(LOCATION(*))
   STATUS(INACTIVE);
   ```

   Change only backups and archives that are at the vault to active status with the following commands:

   ```
   UPDATE ARCHIVE_FILE(*)
   FILTER_BY(LOCATION(VAULT))
   STATUS(ACTIVE);
   UPDATE BACKUP_FILE(*)
   FILTER_BY(LOCATION(VAULT))
   STATUS(ACTIVE);
   ```

4. If you are testing locally, restore the VSAM files using RUV’s NEW_NAME keyword. If you are testing at a disaster recovery site, use of the NEW_NAME keyword is optional. Run the application tests against the newly restored and renamed files.
Phase 2—Performing Disaster Recovery

This section describes how to perform disaster recovery in a real or simulated scenario. You perform the procedures in this section at the disaster recovery site (or at the home site in your disaster recovery testing environment) after the OS/390 system resources, including user catalogs, have been restored and the OS/390 system has been started successfully.

The preparations discussed in Phase 1 are essential to the success of Phase 2; therefore, make sure that all steps in the Phase 1 procedures have been performed before you attempt Phase 2 procedures.

Performing disaster recovery consists of the following procedures:

1. restoring the RUV repository and starting RUV
2. understanding the potential for lost data after a recovery

Restoring the RUV Repository and Starting RUV

To restore the RUV repository, perform the following steps:

1. Define the RUV repository, and start the RUV subsystem.

2. Register any RUV backups or archives that have been created since the time of the repository backup.

   - If you need to change volume serial numbers or SMS classes for VSAM files, define empty VSAM files as necessary and restore the files. RUV will use current definitions as found at the remote site. This makes it fairly simple to change the volume serial numbers or SMS-managed classes.

   - If you do not need to change volume serial numbers or SMS classes for VSAM files, delete the VSAM files to be recovered from the disaster site catalogs.

   - If recovery is to be to a particular point in time, modify the RUV command to reflect the correct recovery point. When no point in time is specified, RUV will recover the files to the most current time possible.

\~\texttt{RESTORE REPOSITORY BACKUP\_IN(DDIN);}
4 Run the restore or recovery jobs as required. RUV will automatically restore files if they are not found in the z/OS catalog.

Understanding the Potential for Lost Data After Recovery

The following information may be lost when a repository is restored:

- any changes made since the repository backup
- the value of the <+1> variable
- registrations of subsequent repository backups

To restore this information to current using RUV, you must journal the repository changes to z/OS system management facility (SMF) logs and use the RECOVER REPOSITORY FORWARD command. See “Recovering an RUV Repository Using SMF Data” on page 568.

Alternatively, you may manually register archives and backups that occurred after the repository backup. You must use the SET <+1>(nn) command to change the <+1> variable to the desired value.

**NOTE**

RUV repository backups cannot be registered. Repository backups must be manually tracked and purged.
Phase 3—Evaluating the Disaster Recovery

The final phase of disaster recovery is evaluation. You need to determine whether the disaster recovery was successful. If it was not successful, you need to determine the reasons for the failure.

Obviously, evaluation is most effective in the context of a simulation; if you find a problem, you can correct it before you run the next simulation. And you can continue to run simulations until you get the results you want. Therefore, by carefully examining the results of your disaster recovery simulations, along with making adjustments in the preparation and performance phases of disaster recovery, you can ensure successful recovery when a real disaster strikes.

You may ask the following questions to evaluate the disaster recovery:

- Were all necessary databases recovered, and were they recovered correctly (without losing too many in-flight transactions)? If not, why did the error occur? The following problems may have caused the error:
  - incorrect or incomplete preparation at the home site
  - unexpected variations at the disaster recovery site

- Was performance acceptable? Did you meet service-level agreements for the length of the database outages? If not, why? The following problems may have caused the unacceptable performance:
  - need for more frequent backups
  - need for more frequent change accumulations
  - need to improve recovery profile options
  - inadequate resources (such as tape drives, CPU cycles, and DASD)
  - need for more recovery points
  - inappropriate backup stacking strategy
  - indexes were recovered rather than rebuilt

You can contact Customer Support to discuss the results of your disaster recovery simulations and to obtain advice for improving these results. Customer Support can help you analyze your preparation procedures and utility parameters and adjust them to ensure data integrity and maximum performance.
Chapter 13 Working with BMC Software Subsystems

This chapter discusses the subsystems that you use with RUV, the BMC Primary Subsystem (BMCP) and the BMC Consolidated Subsystem (BCSS). The following information is included:

BMC Software Subsystems ................................................................. 535
  BMCP ..................................................................................... 535
  BCSS .................................................................................... 536
  Multiple BCSSs ....................................................................... 536
  Subsystem Procedure Names .................................................... 537
  Subsystem Identifiers ............................................................... 537
  Specifying General BMC Subsystem Parameters ......................... 537
  CPU Authorization Passwords .................................................... 538
BMC Software Subsystem (BMCP) ...................................................... 539
  BMCP Command Summary ....................................................... 539
  BMCP Primary Subsystem Parameters ........................................ 540
  Starting the BMCP ................................................................... 541
  Displaying BMCP Status ............................................................ 541
  Stopping the BMCP ................................................................. 542
BMC Consolidated Subsystem (BCSS) .................................................... 543
  BCSS Subsystem ID .................................................................. 543
  BCSS Command Format ............................................................ 543
  Controlling BCSS Using the z/OS Console ..................................... 544
  Using the BCSS Commands Data Set .......................................... 544
  BCSS Command Summary ......................................................... 544
  Starting the BCSS .................................................................... 545
  Specifying BMC Consolidated Subsystem Parameters .................... 545
  Initializing the BCSS RUV Components ....................................... 548
  Displaying the Status of the BCSS Address Space ......................... 549
  Activating the RUV Rule Set ...................................................... 549
  Stopping the RUV BCSS Address Space ....................................... 550
  Requesting RUV Help ............................................................... 550
  Enabling RUV Activity ............................................................. 550
  Stopping RUV Activity Normally .............................................. 551
Stopping RUV Activity Immediately ........................................... 552
Displaying the Status of RUV Components .............................. 552
Changing Ownership of the BWO Coordinator .......................... 553
Displaying RUV Diagnostic Information ................................. 553
Using the VALIDATE SUBSYSTEM Command ......................... 554
Controlling the Subsystem with Subsystem Detail Panel ............ 555
RUV products use two BMC Software standard subsystems for resource management: the BMC Primary Subsystem (BMCP) and the BMC Consolidated Subsystem (BCSS).

A major benefit of using a subsystem architecture is that virtual storage requirements in user address spaces are significantly reduced. The RUV subsystem architecture offers an additional benefit — increased ability to ensure data integrity.

**WARNING**

RUV products can share BMC Software subsystems with other BMC Software products. Some commands discussed in this chapter might affect those BMC Software products. For more information, see the documentation for those products.

The BMCP and the BCSS are maintained independently of the RUV products; their maintenance levels do not correspond to the maintenance levels of the RUV products. When you install a new or updated product that uses the BMCP and the BCSS, use the Installation Check program to determine the level of the existing subsystems against the level of the subsystems from the distribution tape. For RUV products, always use the highest level of subsystem code available.

**BMCP**

The BMCP establishes supervisory services for the BCSS, RUV, and many BMC Software products. It allows interception of open, close, attach, and link requests in the system.

Several products share the BMCP; however, you can have only one copy of the BMCP on a z/OS system.

**NOTE**

If you are running the BMC Software DATA ACCELERATOR Compression product, the BMCP is already installed.

All associated products continue to operate normally even if the BMCP terminates; however, it is recommended that you leave the BMCP running at all times.
BCSS

The BCSS manages I/O to the registration data sets (collectively called the REGISET or repository), manages APF-authorized functions, and performs processing for intercepted open, close, attach, and link requests.

The BCSS performs most of the I/O to the repository during initialization and termination of the step. A small amount of I/O is performed during application program execution.

The BCSS must be active on the z/OS system where you want to execute application programs that use RUV and where you want to access records in the repository through the ISPF interface. You can use the RUV Status Check utility RUVZVML to ensure that required RUV products, components, and functions are available for application execution.

Multiple BCSSs

More than one BCSS can be active on a z/OS image. Non-RUV products, such as DATA ACCELERATOR, can share a BCSS (and the repository) with RUV products or use a separate BCSS (and repository). The BMCP ensures that each BCSS receives control at the proper time for the products using that BCSS.

Whether you use the same BCSS or different ones depends mainly on how you want to manage repository access. To isolate the I/O effects of the products, use multiple BCSS subsystems. If the repository activity is not an issue, multiple products can use the same BCSS.

You must use the MODE parameter in the startup procedure to designate one (and only one) BCSS on a z/OS image as the public BCSS. All others must be designated as private subsystems. The BMCP allows the public BCSS first access to an intercepted request.

A default BCSS can be designated for all RUV products on each z/OS; this default is for use in migration from an earlier version of RUV products. The BCSS identifier (ssid) identifies the subsystem (and repository) in commands and interfaces that work with the BCSS.
Subsystem Procedure Names

The RUV Install System provides two subsystem procedures, one to start the BMCP (default procedure member name BMCP) and one to start the BCSS (default procedure member name BCSSRUVISION$. Each procedure contains the required parameter and statements for the subsystem. If either default name conflicts with the name of another procedure member, you can use a different member name during installation.

**NOTE**

SYS1.PROCLIB is not usually a shared procedure library. If the procedure name and the subsystem ID are identical (which they are if you use the installation defaults for the BMCP) and the procedure is in a non-shared library (such as SYS1.PROCLIB), z/OS runs the subsystem under the master scheduler. After the first attempt to start the procedure, any subsequent attempts to start it result in a “procedure not found” error unless you specify the SUB=jesid parameter (jesid is the JES2 subsystem ID) on the subsystem start command. For example, use the following command to start the BMCP if your JES2 subsystem ID is JES2:

`S BMCP, SUB=JES2`

Subsystem Identifiers

You must use a four-character name to identify the BMCP and the BCSS to z/OS. The RUV Install System uses the default names BMCP and BCSS. If either name conflicts with a non-BMC Software subsystem installed on your system, you can use the SUBSYSID parameter in the BMCP or BCSS startup procedure to change the subsystem names. If you change a subsystem name after the subsystem starts, you must perform an IPL.

Specifying General BMC Subsystem Parameters

There are two types of BMC Subsystem parameters, required and optional. Unless a required parameter is not specified or incorrectly specified, there is no difference in the processing of these options. An operator PROMPT is issued regardless of the NOPROMPT setting.

The BMC subsystem offers many parameters to tailor the operation and function of the subsystem. For example, you can cause a reload of executable modules in common storage to effect maintenance changes without an IPL.
You can select parameter values in the following ways:

- specify them in the PARM field of the procedures JCL EXEC statement for the BMC subsystem start up
- specify them in the actual START command during BMC subsystem start up.

You can specify the parameters in any order. If a conflicting parameter is requested like PROMPT and NOPROMPT then the last (or right-most) parameter is utilized.

If two or more parameters are specified, the list of parameters must be enclosed in single quotation marks or parentheses. Commas must separate each parameter. The parameter list can be no longer than 100 characters, including commas.

**CPU Authorization Passwords**

For RUV products, the BCSS validates CPU authorization passwords. For more information about CPU authorization, see the *Recovery Utility for VSAM Installation and Customization Guide*. 
BMC Software Subsystem (BMCP)

The BMCP establishes supervisory services for the BCSS, RUV, and other BMC Software products. Because the BMCP is shared among several products, only one copy of the BMCP should exist in your system.

The BMCP uses the four-character name BMCP to identify itself to z/OS. This subsystem name is used in all commands and as a suffix in all messages issued by the BMCP. If you are using the subsystem name BMCP with a non-BMC Software subsystem installed in your system, you can change it by modifying the SUBSYSID parameter in the BMCP start procedure. If you change the subsystem name after the BMCP starts, you must perform an IPL for the new name to be recognized by z/OS. See the Recovery Utility for VSAM Installation and Customization Guide for more information.

The BMCP recognizes commands prefixed with the subsystem name followed by at least one blank (for example, BMCP STATUS).

If you are not using the default subsystem names, replace BMCP in each command with the identifier used at your site.

BMCP Command Summary

Table 21 provides a summary of BMCP commands as issued from the z/OS console.

<table>
<thead>
<tr>
<th>RUV BMCP Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCP STATUS</td>
<td>displays the status of the BMCP</td>
</tr>
<tr>
<td>BMCP SHUTDOWN</td>
<td>shuts down the BMCP in an orderly manner by terminating the BMCP address space and BMCP command processing</td>
</tr>
</tbody>
</table>
BMC Primary Subsystem Parameters

This section describes each of the general parameters that can be used to control the BMC primary subsystem. These parameters are provided through either execution parameters or an operator response. When the subsystem is restarted, the previous value for “SUBSYS,” “SVCNO,” and “SMFRECID” are used unconditionally. An IPL is required to change these values. There can be only one primary subsystem active on a z/OS system at any time. All consolidated subsystems use the services of the primary subsystem.

REFRESH  (Default value) During subsystem startup, modules currently loaded into common storage are to be checked for upgrades. If a different version of the module is found in the normal system search, then the module is reloaded into common storage. This option can be used to load new versions of common storage modules without an IPL.

**NOTE**
Not all common storage modules can be upgraded with the REFRESH parameter. For modules that cannot be upgraded using REFRESH, Product Support will issue a statement stating that the upgrade requires an IPL.

DEBUG  The DEBUG parameter indicates that the subsystem should write out diagnostic trace information to the system logs. This should be used with caution to avoid using heavy console traffic.

NODEBUG  (Default value) In NOBUG mode, only normal processing and error messages are issued; no diagnostic messages are issued. This mode is the normal operation mode of the BMC subsystems.

PROMPT  The operator is prompted to correct errors or provide required parameters if they are omitted from the parameter lists.

CAPS  (Default value) Console messages are issued in all upper case letters. This mode will satisfy the CUA language support for console operations as defined by IBM.

MIXED  Console messages are issued in mixed case mode making them easier to read. International language sites may not be able to view the entire BMC message due to DBCS constraints.

SMFRECID  (0) For BMC products that request the BMCP to create a BMC common SMF record, SMFRECID value will be the SMF record id utilized in the SMF records. This value must be a valid user SMF value as defined by IBM. This value is specified as a decimal range between 200 and 255. It should not be utilized by any other products.
SUBSYSID  (BMCP) SUBSYSID specifies the subsystem name utilized during the startup and operation of the primary subsystem. The name must be a unique subsystem name on the z/OS image and cannot exceed four characters. The first character must be alphabetic. For example, "SUBSYSID=BMCP", sets the subsystem name to BMCP.

SVCNO  SVCNO specifies the SVC number utilized by the primary subsystem to service z/OS services intercepts. This value should be changed only if you are instructed to change it by Product Support.

Starting the BMCP

From a z/OS console, enter the following command:

```
S BMCP
```

When you start the BMCP, the following message is displayed on the z/OS console:

```
BMC10100I Subsystem ready. BMCP
```

Displaying BMCP Status

To determine whether the BMCP is active, issue the following command at the z/OS console:

```
BMCP STATUS
```

If the BMCP is active, the following message is displayed on the z/OS console:

```
BMC10700I STATUS,Subsystem ASID(asid) is active  BMCP
BMC10700I STATUS,Debug mode is debug_status  BMCP
```
Stopping the BMCP

Normally, you should leave the BMCP running. If you are applying maintenance to the BMCP, you must terminate the BMCP to install the new modules. You can also terminate the BMCP as part of an orderly shutdown of z/OS before an IPL.

To terminate the BMCP address space, issue the following command at the z/OS console:

**BMCP SHUTDOWN**

Command processing terminates for the BMCP when the BMCP address space ends.

When you shut down the BMCP, the following messages are displayed on the z/OS console:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC101006I</td>
<td>Subsystem termination initiated BMCP</td>
</tr>
<tr>
<td>BMC101228I</td>
<td>Subsystem termination completed BMCP</td>
</tr>
</tbody>
</table>
BMC Consolidated Subsystem (BCSS)

The BMC Consolidated Subsystem (BCSS) provides services for RUV components.

Using BCSS commands at the z/OS console, you can enable the Batch Journaling Facility (the logging functions) and the Backup-While-Open (BWO) feature of RUV. You can also use the z/OS console or the RUV ISPF user interface to control and monitor these processes.

**NOTE**
The BMCP and the BCSS are required to implement and use the Batch Journaling Facility and BWO feature.

BCSS Subsystem ID

BCSS uses the default four-character name BCSS to identify itself to z/OS. All messages issued by BCSS use the subsystem name as a suffix. You can change the subsystem name by changing the SUBSYSID parameter in the BCSS procedure.

BCSS Command Format

BCSS commands can have either of the following formats:

```
ssid action
ssid RUV action [ option ]
```

*ssid* identifies the subsystem identifier of the target BCSS

*RUV* indicates that the command is targeted to the RUV components of the subsystem

*action* identifies the action that will be taken in response to the command

*option* specifies a modification to the command
Controlling BCSS Using the z/OS Console

BCSS recognizes commands that begin with its specified subsystem name followed by at least one blank (for example, ssid STATUS). If you are not using the default subsystem names, replace BCSS in each command with the identifier used at your site. You can use optional command suffixes to specify the scope of the command.

Using the BCSS Commands Data Set

The BCSS commands data set is created during product installation and contains commands that are automatically issued after BCSS startup. BCSS commands and non-BCSS commands can be included in the data set. Non-BCSS commands include all of the z/OS console commands. Commands execute sequentially when the BCSS is initialized and prepared for execution.

BCSS Command Summary

Table 22 lists the RUV BCSS commands that you can use with RUV.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssid REINIT RUV</td>
<td>reinitialize the RUV components of the identified BCSS and load any modules that are new or changed from the subsystem STEPLIB into CSA</td>
<td>548</td>
</tr>
<tr>
<td>ssid RUV DEBUG</td>
<td>display internal diagnostic information</td>
<td>553</td>
</tr>
<tr>
<td>[ BWOMSGLn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ BWODELNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ BWOSETAB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ BWOSETRC ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssid RUV DISABLE</td>
<td>immediately terminates all current Batch Journaling Facility and BWO processes that are in-flight and prevents any new processes from being started</td>
<td>552</td>
</tr>
<tr>
<td>[ ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ LOGGING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ BACKUP ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssid RUV ENABLE</td>
<td>allows Batch Journaling Facility and BWO processes to start again after a DISABLE or QUIESCE command</td>
<td>550</td>
</tr>
<tr>
<td>[ ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ LOGGING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ BACKUP ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssid RUV HELP</td>
<td>displays a short list of RUV subsystem commands on the z/OS console</td>
<td>550</td>
</tr>
<tr>
<td>ssid RUV QUIESCE</td>
<td>allows all current Batch Journaling Facility and BWO processes to finish normally but prevents any new processes from being started and prevents rule sets from being activated</td>
<td>551</td>
</tr>
</tbody>
</table>
You must start the BMCP before you start the BCSS. The BCSS should be started before you start any applications. To start the BCSS, issue the following command at the z/OS console (where bmcbcss is the procedure library member that contains the BCSS procedure):

```
S bmcbcss
```

When you start BCSS, the following message is displayed on the z/OS console:

```
BMC11100I SUBSYSTEM READY ssid
```

### Specifying BMC Consolidated Subsystem Parameters

This section describes each of the general RUV parameters that can be specified to control the BMC consolidated subsystem.

**REFRESH**

(Default value) During subsystem startup, modules currently loaded into common storage are to be checked for upgrades. If a different version of the module is found in the normal system search, then the module is reloaded into common storage. This option can be used to load new versions of common storage modules without the need for an IPL.

---

**NOTE**

Not all common storage modules can be upgraded by the use of the REFRESH parameter. For modules that cannot be upgraded using REFRESH, Product Support will issue a statement stating that the upgrade requires an IPL.
NOREFRESH
During subsystem startup, modules currently loaded into common storage will not be checked for upgrades.

DEBUG
The DEBUG parameter indicates that the subsystem should write out diagnostic trace information to the system logs. This parameter should be used with caution to avoid causing heavy console traffic.

NODEBUG
(Default value) In NOBUG mode, only normal processing and error messages are issued; no diagnostic messages are issued. This mode is the normal operation mode of the BMC subsystems.

PROMPT
(Default value) The operator is prompted to correct errors or provide required parameters if they are omitted from the parameter lists.

NOPROMPT
The operator is prompted only if a required parameter is omitted.

CAPS
(Default value) Console messages are issued in all upper case letters. This mode will satisfy the CUA language support for console operations as defined by IBM.

MIXED
Console messages are issued in mixed case mode, making them easier to read. International language sites may not be able to view the entire message due to DBCS constraints.

SKIPSYN
During BMC subsystem startup, the BCSS will skip the synchronization of the REGISETs if:

- The REGISETs are currently being managed by any other active subsystem in a shared DASD environment.
- This is the first start of the BCSS during the current IPL.

If there are no other BCSS subsystems managing the REGISETs or this is a restart of the BCSS, the files will be synchronized during startup. The impact of the synchronization reserve can be minimized by the use of the SKIPSYN parameter. When both conditions are met, the synchronization process is bypassed and message BMC11058I is issued.
FORCESYN
(Default value) During BMC subsystem startup, the BCSS will synchronize the
REGISETs to ensure that the files are all validated. During the startup the BCSS
will RESERVE the volume that the REGISETs reside on and no other BCSS
subsystem can access the REGISETs until synchronization is completed.

PRIVATE
(Default value) The BCSS subsystem loads an independent set of control blocks
and intercept routines. There is no checking for duplicate functionality. Those
products that request this mode should use this option. Additional BCSS
subsystems can be started to test new code releases; however, care should be
taken to ensure that BCSS services are invoked only once per request.

PUBLIC
The BCSS subsystem loads a common set of control blocks and intercept
routines. There can be only one PUBLIC BCSS subsystem. If more than one
consolidated subsystem is started as PUBLIC, an error message is issued and
the start of the consolidated subsystem will fail. The PUBLIC subsystem
receives control from the BMC intercept routines first, regardless of BMC
subsystem start order. Some BMC products require that they be operational in
only the PUBLIC subsystem.

SUBSYSID
(BCSS) SUBSYSID specifies the subsystem name used during the startup and
operation of the consolidated subsystem. The name must be a unique
subsystem name on the z/OS image and cannot exceed four characters. The
first character must be alphabetic. For every instance of the consolidated
subsystem started within a system, a unique name must be used. For example,
“SUBSYSID=BCSS” sets the subsystem name to BCSS.

CASPROC
(NO DEFAULT) The Compression Address Spaces (CAS) are internally
managed by the consolidated subsystem. When required, the consolidated
subsystem automatically starts a CAS. This parameter notifies the consolidated
subsystem of the PROC name for the CAS. This parameter is required and has
no default value. For example, “CASPROC=BMCCAS” sets the CAS procedure
names to BMCCAS

DSPSIZE
(1) This parameter notifies the consolidated subsystem how large the auxiliary
DATASPACE should be allocated to cache REGISET entries. This value is
specified as megabytes the default is 1, which is large enough to hold
approximately 1,000 REGISET entries. If the REGISET is expected to be larger,
then the DSPSIZE should be adjusted. The cache is managed in an LRU
manner. For example “DSPSIZE=1” sets the DATASPACE size allocated to one
megabyte.
NOREGISET
NOREGISET specifies that the subsystem is to operate without a REGISET (registration data set). During the startup of the subsystem, the initialization routines check to ensure that all required data sets are available for use. If you plan to operate without any registration data sets, this parameter causes the initialization routines to bypass checking those required data sets. Most products that use the BCSS require the use of the REGISETS. Products that require the use of REGISETS will not operate correctly without REGISET availability. The products that can operate without the REGISET will specify this option to be utilized.

NODUPLEX
NODUPLEX specifies that the subsystem is to operate without a DUPLEX registration data set. During the startup of the subsystem, the initialization routines check to ensure that all required data sets are available for use. If you plan to operate with only a PRIMARY registration data set, this parameter causes the initialization routines to bypass the checking for the DUPLEX registration data sets.

Initializing the BCSS RUV Components

Use the REINIT RUV command to initialize and start the RUV components of the BCSS. This command reinitializes the components and loads any modules that are new or changed from the subsystem STEPLIB into CSA. An initialized RUV subsystem is required for access to the RUV repository.

This command is the normal way to apply maintenance without shutting down and restarting the BCSS. Only new processing is affected; all currently active Backup-While-Open (BWO) or Batch Journaling Facility processes continue to use the old modules.

**NOTE**
This command is executed once in the initialization command stream, and in normal production should not have to be reissued. For this reason, there is no provision for removing modules from CSA without an IPL.

To issue the command at the z/OS console, use the following format:

```plaintext
ssid REINIT RUV
```
Displaying the Status of the BCSS Address Space

Use the STATUS command to display the status of the BCSS address space.

To issue the command at the z/OS console, use the following format:

```
ssid STATUS
```

If the BCSS is active, the following status message is displayed on the z/OS console:

```
BMC11700I STATUS, PRIVATE SUBSYSTEM ASID(0074) IS ACTIVE ssid
```

Multiple subsystem service tasks are also active in the BCSS address space. A message similar to the following is displayed for each service:

```
BMC11700I STATUS, TASK(xx), NDISP(xxxxx), SUSPENDED ssid
```

The following message shows the number of service requests queued and not yet processed by BCSS:

```
BMC11700I STATUS, nnnn SERVICES REQUESTS QUEUED ssid
```

The following message shows the number of compression address spaces available to BCSS:

```
BMC11700I STATUS, nnnn COMPRESSION ADDRESS SPACES ssid
```

Activating the RUV Rule Set

Use the RUV ACTIVATE RULE_SET command to activate name32 rule set.

To issue the command at the z/OS console, use the following format:

```
ssid RUV ACTIVATE RULE_SET(name32)
```

You can include the optional TRACE_ON keyword on the RUV ACTIVATE RULE_SET command to view rules as you test them with the TRY command. The TRACE_ON keyword has no values. For more information about the TRY command, see “Testing a Rule Set” on page 201.
Stopping the RUV BCSS Address Space

Use the SHUTDOWN command to terminate the RUV BCSS address space. Normally, you should leave the BCSS running. If you are applying maintenance to the BCSS, you must terminate the BCSS to install the new modules. You can also terminate the BCSS as part of an orderly shutdown of z/OS before an IPL. Command processing terminates for BCSS when the BCSS address space ends.

**WARNING**
*Use the SHUTDOWN command with extreme caution!* Other products might be using the BCSS. Shutting down the BCSS affects all products that are using that BCSS. RUV is unable to protect your data if RUV is shut down. Active logs and journals will be incomplete, and they cannot be registered in the RUV repository because it is unavailable when RUV is shut down.

Before you shut down the BCSS, you can use the RUV Job Monitoring panel (Figure 99 on page 400) to determine which RUV jobs are currently using the BCSS.

To issue the command at the z/OS console, use the following format:

```
ssid SHUTDOWN
```

When you shut down the BCSS, the following messages are displayed on the z/OS console:

```
BMC11106I Subsystem termination initiated ssid
BMC11228I Subsystem termination completed (EOT) ssid
```
Use the following format to issue the command at the z/OS console (ssid is the subsystem identifier of the BCSS):

```
ssid RUV ENABLE [ALL | LOGGING | BACKUP]
```

One of the following keywords is required:

**ALL**
Enable all components.

**LOGGING**
Enable logging components (the Batch Journaling Facility).

**BACKUP**
Enable backup components (the BWO feature).

---

### Stopping RUV Activity Normally

Use the RUV QUIESCE command to stop RUV activity in the RUV in an orderly fashion. This command allows all current Batch Journaling Facility and BWO processes to finish normally but prevents any new processes from being started. Also, rule sets cannot be activated.

---

**WARNING**

RUV may be unable to protect your data if RUV is quiesced. Currently active BWO and Batch Journaling Facility processes that are in-flight will complete normally, but new processes will not be started. Also, an attempt to activate a job-level rule set when logging is quiesced will cause the job to be abended.

To issue the command at the z/OS console, use the following format:

```
ssid RUV QUIESCE [ALL | LOGGING | BACKUP]
```

One of the following keywords is required:

**ALL**
Enable all components.

**LOGGING**
Enable logging components (the Batch Journaling Facility).

**BACKUP**
Enable backup components (the BWO feature).
Stopping RUV Activity Immediately

Use the RUV DISABLE command with extreme caution to stop activity in the RUV components immediately. The command causes an immediate end to all current Batch Journaling Facility and BWO processes that are in-flight and prevents any new processes from being started.

**WARNING**

RUV is unable to protect your data if the RUV components of the subsystem are disabled. The RUV DISABLE command causes active logs and journals to be terminated immediately. The incomplete logs and journals cannot be registered in the RUV repository. It is recommended that you use *extreme caution* in using this command.

To issue the command at the z/OS console, use the following format:

```
ssid RUV DISABLE [ ALL | LOGGING | BACKUP ]
```

One of the following keywords is required:

- **ALL**
  
  Disable all components.

- **LOGGING**
  
  Disable logging components (the Batch Journaling Facility).

- **BACKUP**
  
  Disable backup components (the BWO feature).

**Displaying the Status of RUV Components**

Use the RUV STATUS command to display the status of the RUV components of the BCSS.

To issue the command at the z/OS console, use the following format:

```
ssid RUV STATUS [SECURITY]
```
Changing Ownership of the BWO Coordinator

Use the RUV TAKEOVER BWOCOORD command to change ownership of the BWO coordinator. In this command, ssid is the identifier of the subsystem ID of the subsystem to become the new coordinator:

This command may be needed if the BWO coordinating subsystem terminates or is shut down or has had some other abnormal process interrupt it. In a normal production environment, you would not need to issue this command. You should use it only at the direction of Product Support or after discussing the effects of its use with Product Support.

To issue the command at the z/OS console, use the following format:

```
ssid RUV TAKEOVER BWOCOORD
```

Displaying RUV Diagnostic Information

Use the RUV DEBUG command to display diagnostic information about the RUV components of the subsystem and the Backup-While-Open (BWO) feature.

Use the following format to issue the command at the z/OS console (ssid is the subsystem identifier of the BCSS):

```
ssid RUV DEBUG [BWOMSLn | BWODELNT | BWOSETRC | BWOSETAB]
```

You can use the following keywords to set options to use with the command:

**BWOMSLn**

Use this option to set the level of messages to produce from the BWO feature. Substitute a single digit (1 through 9) for n to display progressively more detail and volume of message tracing. A value of BWOMSL0 is the normal operating mode and the default. It displays only critical error messages and is intended as the normal production mode of operation.

**WARNING**

On a relatively busy system performing several BWO tasks, a debugging message level as low as BWOMSL1 can produce numerous messages to the master console. This option should only be used under the direction of Product Support.
Using the VALIDATE SUBSYSTEM Command

You can use the VALIDATE SUBSYSTEM command of the RUVZSM0 program at the request of Product Support. The command provides information about the subsystem. The command can also create a dump of the RUV control blocks that are in use.

To use the command, use the following syntax:

```
VALIDATE SUBSYSTEM(ssid)
   { DUMP(NO | YES) }
;
```
Controlling the Subsystem with Subsystem Detail Panel

You can monitor and control the BCSSs from the command line as explained earlier. The same functions are available using the RUV ISPF interface. On the Recovery Utility for VSAM Options Menu (see Figure 19 on page 115), select option 8 (Subsystem / Ruleset Information) to display the RUV Subsystem Detail panel (Figure 146). This panel allows you to display information about the subsystem and to issue commands against the subsystem.

Figure 146 Subsystem Detail Panel

The Subsystem Detail panel consists of four areas. The top area displays the RUV product version, subsystem version, and the operating system version, followed by the subsystem ID (SSID) of the subsystem currently connected and the system ID (SMFID) of the system where the subsystem is running.

The second area from the top displays the currently active rule set at the subsystem level. N/A is displayed if no rule set is active.

The third area from the top displays the current state of BACKUP and LOGGING components in the connected subsystem. Selection enables the File/Enable, File/Disable, File/Quiesce and File/Status action bar choices if changes are authorized for the user ID by the security system. These selections are only available when your authorized View is set to Update Mode.
The bottom area provides access to the subsystem QUIESCE command for authorized users of the connected subsystem. Commands are only available when an authorized View is set to Command Mode.

In addition to the standard choices, the following choices are available from the File action bar choice:

**Enable**
- Enter the subsystem ENABLE command. For more information, see “Enabling RUV Activity” on page 550.

**Disable**
- Enter the subsystem DISABLE command. For more information, see “Stopping RUV Activity Immediately” on page 552.

**Quiesce**
- Enter the subsystem QUIESCE command. For more information, see “Stopping RUV Activity Normally” on page 551.

**Status**
- Enter the subsystem STATUS command. For more information, see “Displaying the Status of RUV Components” on page 552.
This chapter provides essential information about the collection of registration data sets—the REGISET—that RUV uses as a repository for control information. This chapter contains the following information:

REGISET Structure and Use .......................................................... 559
  Registration Data Sets ....................................................... 559
  Primary Registration Data Sets ............................................. 559
  Duplex Registration Data Sets ............................................. 560
  REGISET Records ............................................................. 561
  Size of the Repository ....................................................... 562
Registration Data Set Recovery ................................................ 564
  Recovering a Primary Registration Data Set ......................... 564
  Recovery with Another Primary Registration Data Set ............ 564
  Recovery with a Duplex Registration Data Set ....................... 565
  Recovering from a Backup Copy .......................................... 566
  Recovering a Duplex Registration Data Set ......................... 567
  Recovering an RUV Repository Using SMF Data ..................... 568
REGISET Analysis and Backups ................................................ 572
  Backing Up and Restoring RUV Repository Data .................. 573
    Using the BACKUP REPOSITORY Command ......................... 573
    Using the BACKUP_OUT Keyword ..................................... 574
    Using the RESTORE REPOSITORY Command ....................... 574
    Using the BACKUP_IN Keyword ....................................... 574
    Using the MERGE Keyword .............................................. 575
    Coding BACKUP REPOSITORY Statements .......................... 576
    Coding RESTORE REPOSITORY Commands ......................... 576
Printing Repository Data ....................................................... 577
  Using the PRINT REPOSITORY Command ............................... 577
  Using the COUNT Keyword ................................................. 577
  Coding PRINT REPOSITORY Statements ............................... 578
Purging Repository Records .................................................. 579
  Selecting the Purge Technique and Value ......................... 579
  Performing the Purge Procedure ........................................ 580
  Setting the Purge Value at System, Job, and File Levels .......... 580
  Using Commands to Set Purge Values ................................. 581
  Understanding the One-Day Rule ....................................... 581
REGISET Structure and Use

This section explains the physical structure of the REGISET and how it is associated with a BMC Consolidated Subsystem (BCSS). See Chapter 13, “Working with BMC Software Subsystems,” for information about setting up the BCSS.

Registration Data Sets

The REGISET can consist of as few as two (or as many as 22) registration data sets. These data sets are VSAM key-sequenced data sets (KSDSs) that are defined during RUV product installation. The registration data sets that a BCSS uses are identified in the BCSS procedure (default name BMCBCSS) with DD statements.

Registration data sets are of two types—primary and duplex. Primary and duplex registration data sets have identical structures. Whether a registration data set is treated as a primary or a duplex is determined by the ddnames in the BCSS procedure.

Depending on your recovery needs, you can define one of the following scenarios for your site:

- multiple primary registration data sets and no duplex registration data sets
- a single primary registration data set and multiple duplex registration data sets
- multiple primary registration data sets and multiple duplex registration data sets

Each registration data set must reside on a separate volume and cannot be defined as a multivolume data set.

Primary Registration Data Sets

BCSS uses the primary registration data sets when providing RUV components with responses to REGISET access. You must have at least one primary registration data set; you can have as many as 11.

When you define more than one primary registration data set, you are using the Multiple Primary feature. If the active primary registration data set fails, a message is issued, indicating that the failing data set has been dropped from processing. Because an additional primary registration data set is available, normal processing continues, allowing you to recover the failing data set at a more convenient time. In addition, recovery of the failing data set is simplified because restoration is not necessary.
When the BCSS updates a primary registration data set, it issues a hardware reserve against the volume; a primary registration data set should not reside on the same volume with another data set that might be reserved, such as a User catalog, another heavily used VSAM file, or another primary registration data set. The shared reserve does not lock out access from the same z/OS image but does lock out access from other z/OS images in a shared DASD configuration. Reserves are of short duration and occur only during update activity. The reserve is held until updates to all primary registration data sets and all duplex registration data sets are complete. Primary registration data sets are defined with SHAREOPTIONS(4,3).

**WARNING**

Do not convert the hardware reserve to an ENQ.

---

**Duplex Registration Data Sets**

The duplex registration data sets maintain active backup copies of the primary registration data sets. Using this type of registration data set is optional, but you can define as many as 11 duplex registration data sets.

The duplex registration data sets are used only for recovering primary registration data sets. When BCSS updates a primary registration data set, it also updates the duplex registration data sets. It makes these updates one at a time while holding a RESERVE on the primary registration data sets. This ensures that, if a failure occurs in the only available primary registration data set, the duplex registration data sets are, at most, one update behind. Subsystems can continue to function if one primary registration data set is available. Duplex registration data sets are defined with SHAREOPTIONS(4,3).

If you encounter a failure in a primary registration data set and there are no additional primary registration data sets, the BCSS stops. You can restore a primary registration data set by using an additional copy of the primary registration data set, one of the duplex registration data sets, or a backup copy of the primary registration data set.

Although the BCSS does not issue a reserve against the volume that contains a duplex registration data set, do not place a duplex registration data set on the same volume with a primary registration data set.
REGISET Records

During RUV subsystem initialization, RUV initializes the empty REGISETs or uses the primary REGISET to synchronize other REGISETs.

Records are created in the REGISET by using the RUV ISPF interface, by the RUVZSM0 utility, and by the Batch Journaling Facility during application program execution.

Records are deleted from the REGISET by using RUVZSM0 utility commands.

REGISET Serialization Process

The REGISET serialization process in version 1.1.00 of the BMC Software Consolidated Subsystem (BCSS) component permits a RESERVE that was issued under the local QNAME of BMCBCSS to be converted to a global ENQ by your system serialization utility. When all shared REGISET systems are operating at BCSS version 1.1.00 or later, you can convert the RESERVE to a global ENQ without risking data integrity issues, reducing the possibility of developing REGISET drop conditions.

--- WARNING ---

RESERVEs that are issued to the REGISET are frequent and of short duration. It is recommended that you do not allow these RESERVEs to be converted to ENQs. The overhead that is incurred by converting a RESERVE with a disk-based management utility has a negative effect on system performance. CTCA-based products do not exhibit this performance degradation.
Size of the Repository

RUV records numerous actions with control information in the repository. This activity varies greatly from one user environment to another. It is recommended that, after using the system, you then adjust and resize the repository according to your usage and purge cycles.

Calculations for the Data Component

If you want to use a formula to calculate an initial size for the data component of the repository data set, you can use the formula shown in Figure 147. First estimate the number of VSAM files you want RUV to manage, the number of backups that you want to keep, and the number of archives you expect to create.

Figure 147  Formula for Calculating Repository Size

<table>
<thead>
<tr>
<th>Formula entry</th>
<th>Multiplication</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>_____ VSAM files to be tracked</td>
<td>X 2500</td>
<td>= ______ +</td>
</tr>
<tr>
<td>_____ backups to keep X _____ number of days retained</td>
<td>X 1500</td>
<td>= ______ +</td>
</tr>
<tr>
<td>_____ expected archives X _____ number of days retained</td>
<td>X 35000</td>
<td>= ______ +</td>
</tr>
<tr>
<td>100000 typical amount for other records</td>
<td>= 100000 +</td>
<td></td>
</tr>
<tr>
<td>total amount</td>
<td>= ______</td>
<td>= / 4096</td>
</tr>
<tr>
<td>number of 4 KB blocks to allocate</td>
<td>= ______</td>
<td></td>
</tr>
</tbody>
</table>

Backups and archives have a timestamp in the key. This causes the repository to grow as new records are added even when old records are being deleted. Account for the growth by increasing the days retained for backups and archives by the number of days between repository reorganizations. You must periodically reorganize the repository to reclaim the space.
Calculations for the Index Component

To calculate the size of the index component of the repository data set, use the following formula:

\[
( \text{Key Length} / 2 ) \times \text{Data CI/CA Size} \leq \text{Index CI Size}
\]

The following example is based on the following attributes of the data component:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Length</td>
<td>64</td>
</tr>
<tr>
<td>Data CI Size</td>
<td>4096</td>
</tr>
<tr>
<td>Data CI/CA</td>
<td>168</td>
</tr>
</tbody>
</table>

Therefore, the index CI size should be greater than or equal to 8192:

\[
( 64 / 2 ) \times 168 = 5376
\]

Index CI size = 8192

Based on this calculation, if you change the CI size of the data component to 4096, you should change the CI size of the index component to 8192 B (8 KB).

The following Define Cluster command would be used to allocate the repository:

```
DEF CL
   (NAME('RCHRLB.TEST.VSAM.REGISET') -
    SHAREOPTIONS(4,3) -
    REC(6250 ) - defined by # of records
    KEYS(64 0) -
    FREESPACE(10 10) -
    BUFFERSPACE(32768) - half-track buffer size
    RECORDSIZE(4096 4096) - worst case scenario for record size
    SPEED IMBED NOREPLICATE NOREUSE -
    VOL(DEVXXX)) -
   DATA -
      (NAME('RCHRLB.TEST.VSAM.REGISET.DATA') -
       CONTROLINTERVALSIZE(10240)) -
   INDEX -
      (NAME('RCHRLB.TEST.VSAM.REGISET.INDEX') -
       CONTROLINTERVALSIZE(1024))
```
Registration Data Set Recovery

This section provides instructions for recovering a primary or a duplex registration data set.

If the BCSS detects a REGISET failure, it issues a message indicating that a registration data set (primary or duplex) has failed (dropped) and requires recovery. System programmers should recover the registration data set immediately. If you are using multiple primary registration data sets, immediate recovery is not necessary as long as additional primary registration data sets are available for recovery purposes.

Recovering a Primary Registration Data Set

Recovering a primary registration data set is a simple task when an additional primary registration data set is available. If no additional primary is available but a duplex registration data set is available, you can still recover a primary registration data set in a few simple steps.

When possible, use a duplex registration data set rather than a backup copy to recover a lost primary registration data set. If no primary or duplex data sets are available, you can use a backup copy for recovery, but you must perform steps to ensure correct backout and restart of the application programs that use the RUV product.

Recovery with Another Primary Registration Data Set

To recover a primary registration data set when an additional primary registration data set is available, perform the following steps:

1 Required. Issue the following command to stop the BCSS (where ssid is the subsystem identifier):

   /ssid SHUTDOWN

2 Required. Recover the failed primary registration data set. You can delete the failed data set (if it has not already been deleted) and redefine it, or you can define a new data set and change the startup procedure for the BCSS to refer to the new data set instead of the failed data set.

3 Required. Issue the following command to restart the BCSS (where bmcbcss is the procedure library member that contains the BCSS procedure):

   S bmcbcss
Recovery with a Duplex Registration Data Set

To recover the primary registration data set when no additional primary registration data set is available but a duplex registration data set is available, perform the following steps:

1. Required. Issue the following command to stop the BCSS (where ssid is the subsystem identifier):

   `/ssid SHUTDOWN`

2. Required. Delete the failed primary registration data set.

3. Required. Redefine the primary registration data set (the redefined primary becomes the duplex in the next step). Refer to the job in $Q04REPO of the RUV install library.

4. In the BCSS procedure, switch ddname REGISET with ddname DUPLEX. This action causes the valid duplex data set to become the primary and the newly defined data set to become the duplex.

5. Required. Issue the following command to restart the BCSS (where bmcbcss is the procedure library member that contains the BCSS procedure):

   `S bmcbcss`

---

**NOTE**

Perform the steps that involve the BCSS for each BCSS that uses the primary registration data sets that has failed.

---

**NOTE**

This process can take awhile to complete if the repository is very large, as the new primary repository will populate the new duplex repository.
Recovering from a Backup Copy

Because much of the information in registration data sets changes frequently, recover a registration data set from an additional primary or duplex registration data set if at all possible. You can use a backup copy for recovery if each of the following conditions is true, such as in a disaster recovery situation:

- The primary registration data set fails.
- Another primary registration data set is not available.
- No duplex registration data set is available to recover the failed primary registration data set.

After recovery from a backup copy, you must perform steps to ensure correct handling of application program executions that occurred between the backup of the REGISET and at the time of the REGISET failure.

To recover the primary registration data set from a backup copy, perform the following steps:

1 Required. Issue the following command to stop the BCSS (where ssid is the subsystem identifier):
   
   `/ssid SHUTDOWN`

2 Required. Delete the failed primary registration data set.

3 Required. Redefine the primary registration data set. Refer to the job in $Q04REPO of the RUV install library.

4 Required. Use IDCAMS to REPRO the contents of a backup into the redefined primary registration data set.

5 Required. Issue the following command to restart the BCSS (where bmcbcss is the procedure library member that contains the BCSS procedure):
   
   `S bmcbcss`

   _NOTE_
   This process can take awhile to complete if the repository is very large, as the _new_ primary repository will populate the _new_ duplex repository.

6 Register backups and archives, and rerun RUVZSM0 utility jobs that altered the repository content since the backup occurred. Alternatively, see “Recovering an RUV Repository Using SMF Data” on page 568.
Recovering a Duplex Registration Data Set

The BCSS corrects most duplex registration data set failures automatically. When the BCSS cannot fix a failed duplex registration data set, it issues a message stating that you will need to recover a particular duplex registration data set.

To recover a duplex registration data set, complete the following steps:

1  Required. Issue the following command to stop the BCSS (where ssid is the subsystem identifier):

   /ssid SHUTDOWN

2  Required. Delete the failed duplex registration data set.

3  Required. Redefine the duplex registration data set. Refer to the job in $Q04REPO of the RUV install library.

   **NOTE**

   You do not need to use the IDCAMS REPRO command to copy the contents of a primary registration data set into a duplex registration data set. When the BCSS initializes, it makes the necessary changes to ensure the duplex registration data set is an exact copy of the primary registration data set.

4  Required. Issue the following command to restart the BCSS (where bmcbcss is the procedure library member that contains the BCSS procedure):

   S bmcbcss

   **NOTE**

   This process can take awhile to complete if the repository is very large, as the new primary repository will populate the new duplex repository.
Recovering an RUV Repository Using SMF Data

The RUV repository contains important recovery information, including registered backup data and archive data. Some of this recovery information is volatile and difficult to recreate manually.

To assist in recreating a damaged or destroyed repository, RUV provides a mechanism to recover the repository to a point in time through journaling to System Management Facilities (SMF) data sets. You can have RUV gather and place this SMF data into a registered RUV archive. RUV registers the SMF data to a data set name of REPOSITORY_JOURNAL. This task should be done when you set up your RUV Rulesets.

**NOTE**
The repository is unavailable during an ongoing repository recovery process. Repository recovery must performed when no activity is occurring on any VSAM files that RUV monitors.

To avoid potential repository recovery problems, use easily identifiable data set naming standards for all repository archives, including those for SMF data set archives.

To set up RUV subsystems for recovery through SMF journaling, perform the following steps:

1. **(required)** Provide a unique SMF record type / sub-type number to the RUV subsystem by using the following RUV SET DEFAULT command:

```
SET_DEFAULT_SMF_TYPE(nnn {, subtype} );
```

In this command, *nnn* is the SMF record type number and *subtype* is the SMF record subtype number. Valid values for *nnn* are 0 or 128 through 255, and valid *subtype* values are 0 or 1 through 10.

- If *nnn* is set to zero (the default), SMF recovery information is not recorded.
- If *nnn* is set to 128 through 255, SMF recovery information is recorded. The *subtype* value does not have to be set; it defaults to 0.

**NOTE**
Each RUV subsystem that has enabled SMF for recovery must have its own unique SMF record type and subtype combination because RUV uses these numbers to select the SMF data to be restored to any RUV repository.
2 (required) Back up the RUV repository to provide a point-in-time restore point from which to recover by using the following RUV BACKUP REPOSITORY command:

```
BACKUP_REPOSITORY
  BACKUP_OUT(ddn8
    {COMMENT ( comment fields )}
    { LOCATION ( location_name8 )}
    { STATUS ( ACTIVE | INACTIVE )}
  )
  STATUS(ACTIVE)
```

It is recommended that the RUV repository be backed up on a regular basis. You can back up the repository daily or weekly, depending on your site requirements.

3 (optional) Use the following RUV ARCHIVE SMF_IN command set to have RUV pre-process the SMF data to create an archive file containing only the SMF record type number and subtype number for that RUV subsystem:

```
ARCHIVE
  SMF_IN(qsam_SMF_dsn)
  ARCHIVE_OUT(ddn8 )
  COMMENT(comment fields)
  LOCATION(location_name8)
  STATUS(ACTIVE)
  MUST_COMPLETE(YES)
```

$qsam_SMF_dsn$ is the data set name of an SMF archive, such as that produced by the SMF Dump Utility (IFASMFDP). This file may be a general SMF archive that contains all of your site SMF record types, or it may have been filtered by IFASMFDP or another program to contain only the SMF record type that RUV needs to process.

To recover an RUV repository, perform the following steps.

1 (required) Quiesce any work that may be occurring on any of the VSAM files that RUV should be journaling.

2 (required). Identify the RUV repository backup that represents the point in time from which you want to recover which would typically be the latest backup taken of the RUV repository.

3 (required) Identify the SMF dump files or the RUV SMF archives (if these have been created) containing the RUV SMF records for the period since the repository backup identified in step 2. To pick up all RUV SMF records that you want, you may have to dump your current SMF SYSL.MANx files.
(optional) If the repository being recovered has been broken (you are not recovering a repository that is undamaged for business backout reasons), the repository should be recreated by performing the following steps:

A Use IDCAMS to delete and redefine the repository and any alternate primary and duplex data sets. For examples, see the IDCAMS statements in “Exporting and Importing the Repository” on page 592.

B Start the RUV subsystem by using the newly defined data sets.

C Redefine the RUV SET DEFAULT SMF_TYPE parameters that are appropriate for the RUV repository being recovered:

```
SET SUBSYSTEM (ssid);
SET DEFAULT SMF_TYPE(nnn {.subtype});
```

D Reinitialize the RUV subsystem to make the previous SET DEFAULT SMF_TYPE command active by using the system console command:

```
ssid REINIT
```

(required) Recover the repository by using the RUV RECOVER REPOSITORY FORWARD command:

```
RECOVER REPOSITORY FORWARD
{ BACKUP_IN(ddn_dsn) }
{ SMF_IN(ddn_dsn, ...) }
{ ARCHIVE_IN(ddn_dsn, ...) }
{ START_TIME(yyyyjjjhhmmsst) }
{ STOP_TIME(yyyyjjjhhmmsst) }
{ MERGE(REPLACE | NOREPLACE) }
{ SIMULATE }
```

The BACKUP_IN keyword refers to a previous repository backup file that was created by using the RUV BACKUP REPOSITORY command.

The SMF_IN keyword refers to one or more QSAM SMF dump files. These files may be general QSAM SMF dump data sets containing all SMF record types, or they may have been pre-processed by IFASMFDP or a similar program to contain only the desired RUV SMF records. The recovery operation obviously runs faster if only the desired RUV SMF records are present.
The ARCHIVE_IN keyword designates one or more RUV SMF archive files that have been created by using the RUV ARCHIVE SMF_IN command. These files contain only the SMF record type and subtype for the repository being recovered, and should provide the quickest recovery if available. They have a different format from the QSAM SMF dump files, and cannot be used with the SMF_IN keyword.

**NOTE**
Although RUV registers these RUV SMF archives, it does not try to determine automatically the appropriate RUV SMF archives to use for the time period involved for the repository recovery because the contents of the repository cannot be trusted at that time.

Specify the SMF_IN keyword or the ARCHIVE_IN keyword, but not both.

If you do not specify the START_TIME keyword, RUV uses the start time that is recorded on the repository backup (RUV applies RUV SMF records since the time that the backup was taken).

If you do not specify the STOP_TIME keyword, RUV assumes the current time as the stop time (RUV applies all RUV SMF records since the time the backup was taken).

The MERGE(REPLACE | NOREPLACE) keyword functions the same as in RESTORE REPOSITORY.

In the normal case of a broken repository being recovered (after having been redefined) from a backup to the latest point in time, use one of the following commands:

```
RECOVER REPOSITORY FORWARD
 { BACKUP_IN(ddn_dsn) }
 { SMF_IN(ddn_dsn, ...) }
;
```

or

```
RECOVER REPOSITORY FORWARD
 { BACKUP_IN(ddn_dsn) }
 { ARCHIVE_IN(ddn_dsn, ...) }
;
```
REGISET Analysis and Backups

For disaster recovery and other purposes, and if you plan to share the REGISET across CPUs, you should create a daily or weekly job to examine and back up all registration data sets.

To make backup copies of your RUV data in the registration data sets, complete the following steps:

1. Use the JCL in member REGBKUP of the RUV.CNTL library.

   **NOTE**
   
   It is recommended that you create backup copies of the registration data sets with this JCL only. This JCL uses subsystem services to make the backup copies and guarantees the output as being the most recent copies of the registration data sets.

2. Modify the JCL as needed, and submit the job.

   **NOTE**
   
   If RUV is running in the public BCSS and running with BMC products, each individual product must recognize and document procedures specific to their product. The following pages apply only to the RUV data within the REGISET.
Backing Up and Restoring RUV Repository Data

This section describes how to back up and restore the RUV data in the repository with the RUVZSM0 utility.

--- NOTE ---

In RUV reports and commands, the following string identifies a VSAM_FILE record as an RUV repository backup:

```plaintext
--REPOSITORY_JOURNAL
```

For example, to identify the repository backups that are registered, you can execute the REPORT VSAM_FILE command. Then, in the resulting report, search for the string `--REPOSITORY_JOURNAL` to locate the repository backups.

**Using the BACKUP REPOSITORY Command**

Use the BACKUP REPOSITORY command, as shown in the following example, to obtain a copy of all RUV data contained in the repository. You should make this process a regular part of your production file backup process.

```plaintext
BACKUP_REPOSITORY
  BACKUP_OUT(REPBKUP1,
    LOCATION(Local)
    COMMENT(comment_fields)
    STATUS(ACTIVE)
  )
  BACKUP_OUT(REPBKUP2,
    LOCATION(REMOTE)
    STATUS(INACTIVE)
  )
;
```
Using the BACKUP_OUT Keyword

Use the required BACKUP_OUT keyword on the BACKUP REPOSITORY command to identify the ddname of the backup data set to contain the repository data.

You can use the LOCATION keyword with the BACKUP_OUT keyword to assign a location name to this backup data set. Code this keyword after the ddname but before the closing parenthesis of the BACKUP_OUT keyword value. For more information, see “Using the LOCATION Keyword” on page 110.

You can code as many BACKUP_OUT keywords as you need backup data sets.

RUV repository backups are registered but can not be automatically selected. Repository purge processing will automatically delete obsolete repository backups.

Using the RESTORE REPOSITORY Command

Use the RESTORE REPOSITORY command, as shown in the following example, to reset the contents of a repository to an earlier point in time.

RESTORE REPOSITORY
  BACKUP_IN(BKUPREP1)
  ;

You should perform this process only for testing unless no backup or archive processes have occurred since the repository backup was created. You can lose data if you restore an old repository.

Using the BACKUP_IN Keyword

Use the required BACKUP_IN keyword on the RESTORE REPOSITORY command to identify the repository backup data set. Set the value to the ddname in the JCL or the fully qualified data set name.
Using the START_TIME keyword

The START_TIME keyword is valid with the RESTORE REPOSITORY command. You can use this keyword with the STOP_TIME keyword to specify the backup data to restore.

Use the optional START_TIME keyword to specify the start (earliest) timestamp of the range of data to include in the restore process. This timestamp is required if you are using a backup that was not produced by RUV. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the START_TIME Keyword” on page 111.

Using the STOP_TIME keyword

The STOP_TIME keyword is valid with the RESTORE REPOSITORY command. You can use this keyword with the START_TIME keyword to select the range of records to print from the archive.

Use the STOP_TIME keyword to specify the stop (latest) timestamp of the data to include in the restore process. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the RUV ISPF Interface” on page 113.

Using the MERGE Keyword

Use the optional MERGE keyword on the RESTORE REPOSITORY command to combine repository files into a single repository. If the optional MERGE keyword is not specified, then all records on the target repository are deleted before the restore occurs. Records from the backup are merged with the records in the target repository.

The value of the MERGE keyword controls whether RUV replaces duplicate records during the process. Set one of the following values.

REPLACE
   If a duplicate record key is detected, replace the existing record with the record from the backup.

NOREPLACE
   If a duplicate record is detected, retain the existing repository record.
Coding BACKUP REPOSITORY Statements

To code a BACKUP REPOSITORY statement, use the following syntax:

```
BACKUP REPOSITORY
  BACKUP_OUT(ddn8,)
  { LOCATION(locname8) }
  { COMMENT(comment_fields) }
  { STATUS(ACTIVE | INACTIVE) }
}
{ BACKUP_OUT(ddn8,}
  { LOCATION(locname8) }
) ;
```

Coding RESTORE REPOSITORY Commands

To code a RESTORE REPOSITORY statement, use the following syntax:

```
RESTORE REPOSITORY
  BACKUP_IN(ddn8_or_dsn44)
  { MERGE(REPLACE | NOREPLACE) }
;
```
Printing Repository Data

At the request of Product Support, you can use the RUVZSM0 utility to print the data in the repository in hexadecimal format.

--- WARNING ---
Execution of the PRINT REPOSITORY command during a time when the repository is being updated by other processes can produce unpredictable results.

Using the PRINT REPOSITORY Command

Use the PRINT REPOSITORY command, as shown in the following example, to print selected records from the repository:

```plaintext
PRINT REPOSITORY
  COUNT(600,700)
;```

--- NOTE ---
If you are executing the PRINT REPOSITORY command at the request of Product Support, you should execute the SET REPORT(SUMMARY) command first unless otherwise instructed. Summary level reports display repository keys. Detail and full level reports display the complete repository record.

Using the COUNT Keyword

Use the optional COUNT keyword on the PRINT REPOSITORY command to select the range of records to print. Set the value to the relative record number of the first record to print (the from value) and to the relative record number of the last record to print (the to value).
Coding PRINT REPOSITORY Statements

To code a PRINT REPOSITORY statement, use the following syntax:

```
PRINT REPOSITORY
   { COUNT(from, to) }
;
```
Purging Repository Records

RUV automatically registers backups and archives in the repository as they are created. While this information is required to successfully build recovery control job streams, this information needs to be deleted from the repository when the data is no longer needed. You use the PURGE REPOSITORY command to perform this function.

Selecting the Purge Technique and Value

You can control the data to be purged in two ways: by number of days or by number of backups (cycle control). The default value is 45 days. Supported values range from 1 to 7300 days (20 years) of retention.

Forward recovery of a file requires a backup and all archives that were created since the backup was taken. If you take a backup once a week, all journal archives containing data for that file need a minimum retention period of seven days. If the backup is lost or damaged, use the previous backup and specify a retention period of 14 days. You use the DAYS keyword to control the number of days to retain data.

Cycle control uses the number of backups to purge unneeded data. Cycle control requires the use of RUV backups or the registration of other vendor backups. A cycle begins when a backup is created for a VSAM file. All the archive data from the time of this backup becomes part of this cycle, and the next backup starts a new cycle. You use the CYCLES keyword to control the number of cycles to retain.

Be generous with the retention period of backups and archives. If a backup is damaged, you can always deactivate the damaged backup and use the previous backup, with associated archives, to rebuild the VSAM file. Creating and retaining multiple copies of the backup file for sensitive VSAM files is also a good idea.

RUV maintains backup information and the related archive information in the repository. A large number of cycles or number of days increases the number of records in the repository and adds overhead to any processing that involves the repository. Select retention values that protect your data and provide for any historical recovery requirements. As data goes outside the retention periods, it remains in the repository until you execute the PURGE REPOSITORY command. The PURGE command removes the unneeded data from the repository. The PURGE command can optionally produce a QSAM file that lists the data set names of purged backups and archives. This may assist you with management of off-site data.
Performing the Purge Procedure

Use the following procedure to purge the repository:

1. Back up the repository.
2. Build the necessary JCL and PURGE command and options.
3. Run the purge job with the simulate option.
4. Review the output of the simulated purge job, and make adjustments to the purge job if necessary.
5. Run the purge job without the simulate option.
6. Verify the results of the purge job.

Setting the Purge Value at System, Job, and File Levels

You can set the number of days or cycles as a system default, on a per-run basis, or on an individual-file basis. Purge processing selects from the most specific value to the most general as follows.

1. First, RUV checks the individual file for a purge value (in either days or cycles), as set with the DAYS or CYCLES keyword on the ADD VSAM_FILE or UPDATE VSAM_FILE command.
2. If an individual value is not found, RUV checks for a job-level value, as set with the DAYS or CYCLES keyword on the SET command.
3. If no job default is found, RUV checks for a system-level default value, as set with the DAYS or CYCLES keyword on the SET DEFAULT command.
4. If no system default is found, RUV uses a default value of 45 days.

You can mix purging by DAYS or CYCLES within a purge job, but a specific file may use only one purge technique. Different files may have different values.
Using Commands to Set Purge Values

Use the following command to set the default purge value for the subsystem:

```
SET DEFAULT { DAYS(nnn) | CYCLES(nnn) } ;
```

Use the following command to set the purge value for a job step only:

```
SET { DAYS(nnn) | CYCLES(nnn) } ;
```

Use the following command to set a purge value and technique for specific VSAM files:

```
ADD or UPDATE VSAM_FILE(vsam_dsn)
   { DAYS(nnn) | CYCLES(nnn) } ;
```

Understanding the One-Day Rule

RUV uses the one-day rule during purge processing. The one-day rule protects current data from premature deletion. Today is always considered to be day zero. Yesterday is day one. To calculate the retention period, back up to midnight and subtract one day. For example, if you specify DAYS(1), RUV purges backups that were created before midnight yesterday.

Purge Processing Actions

RUV performs the following actions during purge processing:

1. eliminating obsolete APPLID/FILEID combinations
2. purging backups
3. identifying orphan backups
4. examining archives for obsolete data

You may elect to have RUV physically scratch backups and archives from the z/OS catalog when the repository record is deleted.
You may run purges for specific or generic data sets. This feature allows individual users to control purging of their data sets without being concerned with the purge requirements of other users.

The following sections explain each of the actions in more detail and show examples of the reports that are produced as a result of the actions.

**Obsolete APPLID/FILEID References**

The first phase of purge processing checks for obsolete APPLID/FILEID references. If a VSAM DSN is not selected, it is not considered as a purge candidate. An APPLID/FILEID reference is removed if it is more than one day old and points to a VSAM DSN that is no longer in the repository.

*Figure 148* shows an example of the report that RUV produces for purged APPLID/FILEID references.

---

**Figure 148  Purged APPLID/FILEID Reference Report**

```
2009.291 21:10:47 GMT       BMC RECOVERY UTILITY for VSAM -- Version 04.01.00       Page: 2
Applid/Fileid Purge for: VRM1  Simulation mode
Applid: RCHFESA   Fileid: F0000001   VSAM DSN: RUVRUV.HLK.ESDS.PATH1
DSN was selected, VSAM DSN was not found
DSN reference deleted
Applid: RCHFESA   Fileid: F0000002   VSAM DSN: HLK.QA.TEST1.MAIN1
DSN was selected
Applid: RUVQA11I   Fileid: RUVEBASE   VSAM DSN: RUVRUV.DV72IVP.RUVEBASE
DSN was selected, VSAM DSN was not found
Applid: RUVQA11I   Fileid: RUVESMS0   VSAM DSN: RUVRUV.DV72IVP.RUVESMS0
DSN was not selected
```

RUV202157I RUV 1 repository records deleted.
RUV202198I RUV Simulation mode only.
The second phase of purge processing examines registered backups. RUV can purge backups by CYCLES or DAYS. Data backups are RUV and SMS backups. Non-data backups are created by an RBA0 event. RUV does not purge SMS backups. RUV does not delete RBA0 backups until all prior data backups have been deleted.

If you select purging by DAYS and all backups for a file are RUV backups, RUV makes the decision to purge the backups strictly on number of days. Remember, RUV does not delete SMS backups; SMS does. RUV always honors the SMS definition.

If you select purging by CYCLES and all backups for a file are RBA0 backups, purging applies to the RBA0 backups.

If you have a set of mixed data and non-data backups, RUV counts only the data backups as cycles. RUV counts an RBA0 backup as a cycle only when an older data backup does not exist. RUV does not delete SMS backups. If an SMS backup was created before an RUV backup, RUV suspends cycle purging until the SMS backup has been deleted.

**Figure 149** shows an example of the report that RUV produces for purged backups.

**Figure 149  Purged Backup File Report (Part 1 of 2)**

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Event Description</th>
<th>DSName</th>
<th>Purge Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.291 17:02:15 Local</td>
<td>BACKUP REPORT</td>
<td>RUVRUV.DV73IVP.RUVKBASE</td>
<td>CYCLES(4)</td>
</tr>
<tr>
<td>2009.291 22:02:15 GMT</td>
<td>BACKUP REPORT</td>
<td>RUVRUV.DV73IVP.RUVKBASE</td>
<td>CYCLES(4)</td>
</tr>
</tbody>
</table>

Not selected, VSAM DSN: ISIRAM.RUV.TEST.KSDS

Backups for VSAM DSN: ISIRAM.RUV.TEST.KSDS1 Purge method: CYCLES(2)

Purged.... 2009.284 20:29:06 ISIRAM.RUV.TEST.KSDS1.BKUP.V0000059 CYCLES(2)

Purged.... 2009.284 20:31:04 ISIRAM.RUV.TEST.KSDS1.BKUP.V0000067 CYCLES(2)

RUV20184W RUV DSName: ISIRAM.RUV.TEST.KSDS1.BKUP.V0000067 was not found in the z/OS catalog.

Orphaned Backups

The third phase of purge processing identifies and reports registered backups that point to VSAM DSNs that are no longer registered in the repository. These backups are called orphaned backups. You should remove the backup if the file is no longer needed, or you should add the VSAM file to the repository.

Figure 150 shows an example of the report that RUV produces for orphaned backups.

Archives

The final phase of purge processing examines archives. An archive is considered to be obsolete when all data in the archive is no longer needed. Archives can contain data for more than one VSAM file. Archives are not purged from the repository until the recovery data for all files which are represented on the archive has expired.

Figure 151 shows an example of the report that RUV produces for examined archives.
Purge reports can be rather large. You can select from the following levels of reporting (as specified with the SET or SET_DEFAULT REPORT command):

- summary
- detail
- full

RUV displays the target purge date and time if it is performing the purge by DAYS(nn). For example, if you specify the following command:

```sql
SET DAYS(5) REPORT(DETAIL);
PURGE REPOSITORY;
```

RUV reports the following information:

```
Purging data older than: 2006.286 GMT
  Sat Oct 13 00:00:00 2006 GMT
  Fri Oct 12 19:00:00 2006 Local
```
Summary-Level Reports

Summary-level reports display the backup and archive DSNs that are no longer needed. They show only purged data. Figure 152 shows an example of the purge report with summary-level reporting.

**Figure 152  Purge Report with Summary-Level Results**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.291 22:40:59 GMT</td>
<td>Backup File Purge for SSID VRM1  Simulation mode</td>
</tr>
<tr>
<td>2009.291 22:40:59 GMT</td>
<td>Archive File Purge for SSID VRM1  Simulation mode</td>
</tr>
<tr>
<td>2009.291 22:40:59 GMT</td>
<td>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00</td>
</tr>
</tbody>
</table>

RUV202157I RUV 0 repository records deleted.
RUV202198I RUV Simulation mode only.
2009.291 22:40:59 GMT  BMC RECOVERY UTILITY for VSAM -- Version 04.01.00  Report Level: Summary
Applid/Fileid Purge for: VRM1  Simulation mode
Applid/Fileid Sequence

2009.291 22:40:59 GMT
Applid/Fileid Purge for: VRM1  Simulation mode
Applid/Fileid Sequence

RUV202157I RUV 2 repository records deleted.
RUV202198I RUV Simulation mode only.
2009.291 22:40:59 GMT  BMC RECOVERY UTILITY for VSAM -- Version 04.01.00  Report Level: Summary
Applid/Fileid Purge for: VRM1  Simulation mode
Applid/Fileid Sequence

VSAM DSN: ISIRAM.RUV.TEST.KSDDS1.BKUP.V00000059
Purge backup: ISIRAM.RUV.TEST.KSDDS1.BKUP.V00000067
VSAM DSN: ISIRAM.RUV.TEST.KSDDS1
Purge backup: ISIRAM.RUV.TEST.KSDDS1.BKUP.V00000067
VSAM DSN: ISIRAM.RUV.TEST.KSDDS1
Purge backup: ISIRAM.RUV.TEST.KSDDS1.BKUP.V00000067

RUV202157I RUV 2 orphaned backup records.
RUV202198I RUV Simulation mode only.
2009.291 22:40:59 GMT  BMC RECOVERY UTILITY for VSAM -- Version 04.01.00  Report Level: Summary
Applid/Fileid Purge for: VRM1  Simulation mode
Applid/Fileid Sequence

Backup DSN: ISIRAM.ESDS.RL80.BKUP  2009.290 15:47:07

2009.291 22:40:59 GMT  BMC RECOVERY UTILITY for VSAM -- Version 04.01.00  Report Level: Summary
Applid/Fileid Purge for: VRM1  Simulation mode
Applid/Fileid Sequence

Purged: 2009.284 20:02:09 Archive DSN: RUVRRV.ISIRAMTU.V0000045.BLEND
Detail-Level Reports

Detail-level reports add the reason for the purge. They show less data than the full reports; only selected files are displayed. Only purged APLID/FILEIDS are shown in the APLID/FILEID Purge section of the report. The Backup File Purge sections displays only purged backups. The Archive File Purge displays all archives and results of the purge. Figure 153 shows an example with detail-level reporting.

Figure 153  Purge Report with Detail-Level Results

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009.291 22:24:29 GMT</td>
<td>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00</td>
</tr>
<tr>
<td>Applid/Fileid Purge for: VRM1</td>
<td>Simulation mode</td>
</tr>
<tr>
<td>Applid/Fileid Sequence</td>
<td></td>
</tr>
<tr>
<td>RUV20157I</td>
<td>RUV 0 repository records deleted.</td>
</tr>
<tr>
<td>RUV20198I</td>
<td>RUV Simulation mode only.</td>
</tr>
<tr>
<td>2009.291 22:24:29 GMT</td>
<td>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00</td>
</tr>
<tr>
<td>Backup File Purge for SSID VRM1</td>
<td>Simulation mode</td>
</tr>
<tr>
<td>VSAM DSN Sequence</td>
<td></td>
</tr>
<tr>
<td>Purge backup: ISIRAM.RUV.TEST.KSDS1.BKUP.V0000059</td>
<td>CYCLES(2)</td>
</tr>
<tr>
<td>PUR202184W</td>
<td>RUV DSName: ISIRAM.RUV.TEST.KSDS1.BKUP.V0000059 was not found in the z/OS catalog.</td>
</tr>
<tr>
<td>Purge backup: ISIRAM.RUV.TEST.KSDS1.BKUP.V0000067</td>
<td>DAYS(5)</td>
</tr>
<tr>
<td>RUV20157I</td>
<td>RUV 2 repository records deleted.</td>
</tr>
<tr>
<td>RUV20198I</td>
<td>RUV Simulation mode only.</td>
</tr>
<tr>
<td>2009.291 22:24:29 GMT</td>
<td>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00</td>
</tr>
<tr>
<td>Backup File Purge for SSID VRM1</td>
<td>Orphaned Backups</td>
</tr>
<tr>
<td>VSAM DSN Sequence</td>
<td></td>
</tr>
<tr>
<td>Backup DSN: ISIRAM.ESDS.RL80.BKUP</td>
<td>2009.290 15:47:07</td>
</tr>
<tr>
<td>RUV202158W</td>
<td>RUV 2 orphaned backup records.</td>
</tr>
<tr>
<td>2009.291 22:24:29 GMT</td>
<td>BMC RECOVERY UTILITY for VSAM -- Version 04.01.00</td>
</tr>
<tr>
<td>Archive File Purge for SSID VRM1</td>
<td>Simulation mode</td>
</tr>
<tr>
<td>Registration Time Sequence</td>
<td></td>
</tr>
<tr>
<td>Archive DSN: RUVRUV.ISIRAMT2.V0000037.BLEND</td>
<td></td>
</tr>
<tr>
<td>Registration Date: 2009.284 15:44:02 GMT</td>
<td></td>
</tr>
<tr>
<td>ISIRAM.TEMP2</td>
<td></td>
</tr>
<tr>
<td>Archive DSN: RUVRUV.ISIRAMT2.V0000040.BLEND</td>
<td></td>
</tr>
<tr>
<td>Registration Date: 2009.284 15:44:06 GMT</td>
<td></td>
</tr>
<tr>
<td>ISIRAM.TEMP2</td>
<td></td>
</tr>
<tr>
<td>Archive DSN: RUVRUV.ISIRAMTU.V0000043.BLEND</td>
<td></td>
</tr>
<tr>
<td>Registration Date: 2009.284 20:01:21 GMT</td>
<td></td>
</tr>
<tr>
<td>ISIRAM.RUV.TEST.KSDS</td>
<td></td>
</tr>
<tr>
<td>Archive DSN: RUVRUV.ISIRAMTU.V0000045.BLEND</td>
<td></td>
</tr>
<tr>
<td>Registration Date: 2009.284 20:02:09 GMT</td>
<td></td>
</tr>
<tr>
<td>ISIRAM.RUV.TEST.KSDS</td>
<td></td>
</tr>
<tr>
<td>Purged: 2009.284 20:02:09 Archive DSN: RUVRUV.ISIRAMTU.V0000045.BLEND</td>
<td></td>
</tr>
</tbody>
</table>
**Full-Level Reports**

Full-level reports display all VSAM DSNs on each selected archive and a retain/purge reason for each VSAM DSN. Examples with full-level reporting are shown in “Purge Processing Actions” on page 581.

**Purge Output File**

The PURGE command can produce a QSAM file of purged backups and archives. The QSAM file is requested by including the PURGE_OUT keyword in the PURGE command. The DCB defaults for the PURGE_OUT data set are RECFM=VB, LRECL=1596, and BLKSIZE=half track (on a 3390 device, this value is 27998).

The file has variable length records that contain the following information:

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004</td>
<td>RDW</td>
</tr>
<tr>
<td>0008</td>
<td>type of file purged (BACKUP, ARCHIVE, APPLID/FILEID)</td>
</tr>
<tr>
<td>0044</td>
<td>data set name</td>
</tr>
<tr>
<td>0004</td>
<td>count of VOLSERs</td>
</tr>
<tr>
<td>1536</td>
<td>a string of VOLSERs</td>
</tr>
</tbody>
</table>

**Backup before Purge Processing**

Records are permanently removed by the purge process. It is generally a good idea to perform a repository backup prior to the purge process.

**Using the PURGE REPOSITORY Command**

Use the PURGE REPOSITORY command, as shown in the following example, to remove outdated records from the repository.

```
PURGE REPOSITORY
    PURGE_OUT(PURGREPT)
    VSAM_FILE(GNL*)
    VSAM_GROUP(name32, *, ...)
    SCRATCH (NO | YES)
    SIMULATE
    ;
```
Using the PURGE_OUT Keyword

Use the optional PURGE_OUT keyword on the PURGE REPOSITORY command to identify a QSAM data set to contain the data records resulting from the purge process. Set the value of the ddname that identifies the data set in the JCL.

You can use this data for purposes such as auditing. The file and data have no use to RUV.

Using the VSAM_FILE Keyword

Use the optional VSAM_FILE keyword on the PURGE REPOSITORY command to select VSAM file records for purge processing. Set the value to the file name of the records to purge. You can use masking characters in the file name, and you can use multiple values.

Using the VSAM_GROUP Keyword

Use the optional VSAM_GROUP keyword on the PURGE REPOSITORY command to select VSAM file records for purge processing. The VSAM group resolves to the appropriate and individual VSAM files at run time. The definition is not purged.

Using the SCRATCH Keyword

Use the optional SCRATCH keyword on the PURGE REPOSITORY command to control whether RUV issues a request to scratch from the z/OS catalog the data sets that are associated with the purged records. Set one of the following values. The default value is NO.

NO  Do not scratch the data sets.

YES  Issue a request to scratch the data sets.

Using the SIMULATE Keyword

Use the optional SIMULATE keyword on the PURGE command to check for desired results before actually performing a purge. RUV produces standard reports but performs no actual deletions. Records selected and displayed during simulation mode are not written to the ddname specified by the PURGE_OUT keyword.

RUV issues message RUV202198I to indicate that it is performing a simulation. This message follows message RUV202157I. Simulation mode is also indicated in the report page headers.
Coding PURGE REPOSITORY Statements

To code a PURGE REPOSITORY statement, use the following syntax:

```
PURGE  REPOSITORY
  {  PURGE_OUT(ddn8)  }
  {  VSAM_FILE(*, ...)  }
  {  VSAM_GROUP(name32.*, ...)  }
  {  SCRATCH(NO | YES)  }
  {  SIMULATE  }
;  
```
Validating the Repository

Occasionally, invalid repository records may be created when the operating system abnormally terminates or an RUV job is canceled. You can use the VALIDATE REPOSITORY command to detect and, optionally, correct repository errors by deleting invalid records.

Coding the VALIDATE REPOSITORY Command

To code the VALIDATE REPOSITORY command to detect invalid records in the repository, use the following syntax:

```
VALIDATE REPOSITORY
   { SIMULATE }
;```

You can use the optional SIMULATE keyword to produce a report without modifying the repository. This report is useful to determine whether the repository is damaged.

In addition, the VALIDATE REPOSITORY command identifies VSAM files for which delete/define information is not stored in the repository. You can tell RUV to gather delete/define information for these files by using the UPDATE VSAM_FILE command or the UPDATE VSAM_GROUP command.

You should execute the VALIDATE REPOSITORY command during periods of low repository activity.

Improving Repository Performance

To improve the performance of the repository, you can perform a periodic export and import process on the repository. Or you can perform a reorganization process. The export and import process can be faster than the reorganization process. Both processes yield the same results.
Exporting and Importing the Repository

To export and import the repository, perform the following process during a quiet time:

1. Perform a LISTCAT of your primary and secondary repository files. Examine the currently allocated space to ensure that the new space allocation will be adequate.

2. Back up your primary repository by executing the following RUVZSM0 utility command:

   ```
   SET SUBSYS(x);
   REPORT(DETAIL);
   BACKUP REPOSITORY
   BACKUP_OUT(ddn8);
   
   NOTE
   Make sure that the Primary and Secondary repositories are the same size. Also note that the CI size has been changed to 2048 (at the recommendation of IBM).
   ```

   ```
   //EXPORT EXEC PGM=IDCAMS
   //SYSOUT DD SYSOUT=* 
   //SYSPRINT DD SYSOUT=* 
   //OUTDD1 DD DSN=RUV.RUV.REGISET.BACKUP, 
   // UNIT=SYSDA,SPACE=(CYL,(60,13),RLSE), 
   // DCB=(BLKSIZE=40874,LRECL=4087,RECFM=VB), 
   // DISP=(NEW,CATLG,DELETE) 
   //SYSIN DD *
   EXPORT - 
   RUV.RUV.REGISET - 
   OUTFILE(OUTDD1) - 
   TEMPORARY
   //*--------------------------
   //DELETE EXEC PGM=IDCAMS,COND=(3,LT)
   //SYSPRINT DD SYSOUT=* 
   //SYSSIN DD *
   DELETE RUV.RUV.REGISET 
   DELETE RUV.RUV.REGISET2 
   SET MAXCC=0 
   /* 
   //DEFINE EXEC PGM=IDCAMS,REGION=1M
   //SYSPRINT DD SYSOUT=* 
   ```

3. Bring down your RUV subsystem.

4. Execute the following JCL to export your repository data, delete and define your repository data sets, and repopulate your primary repository data set.
5 Start your RUV subsystem again.

The RUV subsystem automatically populates the secondary repository from the primary repository at startup.

6 Perform a LISTCAT of your new primary and secondary repository files.

7 Compare the LISTCAT from before and after the export and import process to assess the results of the process.
Reorganizing the Repository

To reorganize the repository, perform the following process during a quiet time:

1. Perform a LISTCAT of your primary and secondary repository files. Examine the currently allocated space to ensure that the new space allocation will be adequate.

2. Back up your primary repository by using the following RUVZSM0 utility command:

   ```
   SET SUBSYS(x);
   REPORT(DETAIL);
   BACKUP_REPOSITORY
   BACKUP_OUT(ddn8)
   ;
   ```

3. Bring down your RUV subsystem.

4. Delete and redefine your primary and secondary repositories.

   Make sure that the secondary repository is not located on the same volume as your primary repository. Also, make sure that you allocate a secondary extent for the index.

5. Start your RUV subsystem.

6. Run a batch job to recover the repository, as follows:

   ```
   RECOVER_REPOSITORY FORWARD
   BACKUP_IN(ddn_dsn) <=== your backup file from step 2
   ARCHIVE_IN(ddn_dsn_list) <== if needed
   MERGE(REPLACE))
   ;
   ```

   The RUV subsystem automatically restores the primary repository from the backup and populates the secondary repository from the primary repository. This process might take a noticeable amount of time, depending on the size of your repositories.

7. Perform a LISTCAT of your new primary and secondary repository files.

8. Compare the LISTCAT from before and after the reorganization process to assess the results of the process.
Chapter 15 Using VSAM Groups

This chapter introduces the Recovery Utility for VSAM (RUV) VSAM group feature. The following topics are included:

Overview of VSAM Groups .............................................................. 596
  Group Definition ................................................................. 596
  Group Use ........................................................................... 597
  Group Backups ................................................................. 597
  Group Restore ................................................................. 597
  Group Recover ................................................................. 597
  Groups in Rule Sets ........................................................... 598
  Group Maintenance ........................................................... 598
  Group Archives ................................................................... 599
  Group Reports ..................................................................... 600
  Group Purge ....................................................................... 600
  Group Update ...................................................................... 600

Defining Groups ........................................................................ 601
  Using the ADD VSAM_GROUP_DEFINITION Command ......... 601
  Coding the ADD VSAM_GROUP_DEFINITION Command .... 601

Using Groups ........................................................................... 603
  Using the BACKUP VSAM_GROUP Command ....................... 603
  Using the RESTORE VSAM_GROUP Command ..................... 605
  Using the RECOVER FORWARD VSAM_GROUP Command .... 607
  Using the RECOVER UOW VSAM_GROUP Command ............ 609
  Using the RECOVER BACKOUT VSAM_GROUP Command .... 610
  Using the ARCHIVE ACCUM VSAM_GROUP Command ...... 611
  Using the ADD VSAM_SET Command ................................. 613

Maintaining Groups ................................................................. 615
  Using the ADD OUTPUT_MODEL Command ......................... 615
  Using the UPDATE OUTPUT_MODEL Command .................... 617
  Using the DELETE OUTPUT_MODEL Command .................... 617
  Using the REPORT OUTPUT_MODEL Command .................... 617
  Using the UPDATE VSAM_GROUP_DEFINITION Command ....... 618
  Using the DELETE VSAM_GROUP_DEFINITION Command ...... 618
  Using the DELETE VSAM_GROUP Command ....................... 619
  Using the UPDATE VSAM_GROUP Command ....................... 619
  Using the REPORT VSAM_GROUP_DEFINITION Command ....... 620
  Using the REPORT VSAM_GROUP Command ....................... 621
  Using the PURGE REPOSITORY VSAM_GROUP Command .... 621
Overview of VSAM Groups

The RUV VSAM group feature provides a flexible method for working with a collection of RUV-registered VSAM files. This is similar to the way that you work with a single file. After the VSAM files have been registered and a VSAM group has been defined, the VSAM_GROUP keyword can replace the VSAM_FILE keyword in most commands, including BACKUP, RESTORE, RECOVER, and ARCHIVE.

When an RUV command is issued using VSAM groups, the data sets contained in the group resolve to their component Base Cluster VSAM File(s) at run time as registered in the repository. Thus, if you specify VSAM group definition using DSN masking, new VSAM files will be selected automatically as soon as they are registered in the repository.

Figure 154 illustrates an example of VSAM group use.

Figure 154  Group Setup Example
Group Definition

Group definitions are built by identifying data sets to be included or excluded. The VSAM files can be specified by fully qualified DSNs, DSNs with masking, or a combination of both.

The group name is supplied by the user. The group name can have up to 32 characters and is stored in the RUV repository.

NOTE
A group definition without any INCLUDEs would be of no use for any command except rule sets. The order of the statements in the definition does not affect the results of any command.

Group Use

Once a VSAM group is defined, RUV can perform backup, restore, recover and archive operations on multiple files. Group definitions allow the user to grow and change the number of VSAM files managed without JCL changes.

Group Backups

RUV resolves and expands the group into VSAM file names; this expansion is based on the VSAM file names that are recorded in the RUV repository at execution time. The expanded group of files can be backed up in parallel to DASD, multiple tape drives, or stacked tapes, providing you with superior tape utilization.

For RUV to create RUV backups of a group of VSAM files, you must use the OUTPUT_MODEL keyword (see page 604) to supply the data set names of the output files and their allocations. For RUV to create Instant Snapshot copies of a group of VSAM files, you must use the VSAM_NAME_MODEL keyword to specify the VSAM name model to use for constructing the output data set names.

Backup data sets created by VSAM group backups may be used for individual file restore or recovery, as well as VSAM group restore and recovery.
Group Restore

Restoring VSAM groups rebuilds the VSAM file clusters for the files within the group from each file’s current backup. Files in groups are restored in parallel provided that the memory resources and file resources are available.

VSAM group restore operations are not synchronized. The latest backup of each file is used, regardless of the source. For example, SMS backups may be used for some files, while RUV backups can be used for other files in the group. If you want to restore a data set from a backup that is not the current backup, you must use the individual VSAM File restore command, where a BACKUP_IN can be named. Also, you can temporarily update the group definition to exclude the file, restore the group, and then add the file back into the group.

Group Recover

You can use the forward recovery process to rebuild files in a VSAM group. You can also use forward recovery to recover files within a group forward to a point in time, such as the completion of a particular job. The forward recovery process may use backups of the files in a group as well as the after-image journal records to rebuild the file.

Forward or backout recovery of VSAM groups does not provide synchronization between individual files. Therefore, coordination of recovery is limited for all processes specified to a common time. Timestamps can be provided to specify a group’s beginning and ending recovery time points.

Groups in Rule Sets

When VSAM groups are added to the rule set VSAM_SET command, the VSAM group definitions are used. The definitions are rearranged so that the excludes are used before the includes. Selection of data sets is the same as for any other group function. The definitions replace the group name position in the rule set.

NOTE
You are allowed to use the subparameters BOTH, JOURNAL, or LOG only when specifying VSAM groups. These subparameters are used with the INCLUDE definitions.
Group Maintenance

VSAM group maintenance applies to archiving, updating, purging, reporting, and deleting of VSAM files by group. RUV does not allow group maintenance for ADD VSAM_FILE and REGISTER BACKUP_IN commands.

Group Archives

VSAM files defined by a VSAM group definition can be archived and accumulated into consolidated archives or last image archives by group.

When VSAM groups are archived, you can specify whether to retain journal records, log records, or both, and you can consolidate them into a single archive.

As shown in Figure 155, RUV uses a last image process to provide journal record accumulation. During journal record accumulation, this process sorts journal records in file ID, key, and date-time sequence. It then writes a journal record for only the latest (most recent) journal record activity for a record within a file.

This filtering of the most current journal record reduces the size of your journal data set and the time required to perform a forward recovery.

Figure 155  Archive Accumulation Using the Last Image Process
Group Reports

RUV reports can be used to verify the contents of each group. You can request a report that shows the following items:

- the group definition
- files that are in the group
- output environment for the groups.

You can use the INVENTORY_ONLY(YES) parameter to display which files would be used for a particular command without actual processing of file content.

Group Purge

VSAM groups can be used to restrict the PURGE REPOSITORY command to work on only those backups and archives that contain data for files included in the VSAM group.

Group Update

There are two versions of VSAM group updates that are similar in appearance but have dramatically different results.

The first version updates the definition of a group. You may add, update, or delete VSAM group definitions. You must provide a complete definition each time you add or update a group definition. Only the definition is changed; individual files are not affected.

The second version updates information for the individual VSAM files that the VSAM group resolves to at run time. This provides a convenient method to add or alter parameters of all VSAM files in the group. For example, you can add or change the number of days or cycles that control purge processing for a group of VSAM files.
Defining Groups

You define VSAM groups by using the ADD VSAM_GROUP_DEFINITION command of the RUVZSM0 utility.

Using the ADD VSAM_GROUP_DEFINITION Command

Use the ADD VSAM_GROUP_DEFINITION command to create a VSAM group record in the repository.

This group can have a name of up to 32 characters. It may have a comment, but must have at least one VSAM file or DSN mask defining which file(s) with which to work. Each DSN mask may be fully qualified or have wildcards. The mask(s) will resolve to individual VSAM files at each run time using only files registered in the repository.

Coding the ADD VSAM_GROUP_DEFINITION Command

To code an ADD VSAM_GROUP_DEFINITION statement, use the following syntax:

```
ADD VSAM_GROUP_DEFINITION(name32)
| COMMENT(comment_fields) |
VSAM_FILE(dsn44,
  | INCLUDE | EXCLUDE |
  )
VSAM_FILE( ... )
;
```

COMMENT Keyword

The COMMENT keyword is available for use with RUV VSAM group commands and keywords that describe RUV VSAM groups. You can use it for documentation or other purposes to help you control recovery information.

Code the COMMENT keyword as shown in the following example. For the comment value, you can code a maximum of 200 bytes in four 50-byte fields. Enclose each field in quotes if the field contains any spaces. Enclose all fields in parentheses.

```
COMMENT('this is a comment'
  'this is more information'
  'this is a third line of information'
  'this is the last line of the comment')
```
**VSAM_FILE Keyword**

Use the required VSAM_FILE keyword to identify the VSAM data set names to be selected or excluded as members of VSAM_GROUP_DEFINITION. The VSAM_FILE can have masking or be fully qualified data set names. You can select the same VSAM file for more than one VSAM group. Each VSAM_GROUP is completely independent of all other VSAM_GROUP.

**EXCLUDE**

Use the optional EXCLUDE value to prevent a particular VSAM data set from being selected into a VSAM group. For example, if you have a list of VSAM data sets that were selected by using a wildcard, you may want to specify one or more data sets in that list for exclusion from the group.

**INCLUDE**

Use the optional INCLUDE value to select one or more particular VSAM data sets in a group.

---

**NOTE**

EXCLUDE and INCLUDE are mutually exclusive values. During RUV processing of groups, the EXCLUDE value has precedence over the INCLUDE value.
Using Groups

VSAM groups can be used in combination with individual files, depending on your data recovery needs. RUV can use VSAM groups for backup, restore, recovery, and archive operations.

RUV automatically registers backups and archives in the repository as they are created, and this applies to those created from VSAM groups. While this information is required to successfully build recovery control job streams, this information needs to be deleted from the repository when the group data is no longer needed. You use the PURGE command to perform this function.

See Chapter 14, “Maintaining the RUV Repository” for more information on the PURGE REPOSITORY command.

Using the BACKUP VSAM_GROUP Command

Use the BACKUP VSAM_GROUP command to back up multiple VSAM files with a single command. Files will be backed up in parallel provided that memory is sufficient and that the OUTPUT_MODEL keyword provides multiple output devices.

Syntax of BACKUP VSAM_GROUP

The following backup command shows the new parameters introduced for group backup support. To code a BACKUP VSAM_GROUP statement with the OUTPUT_MODEL keyword, use the following syntax:

```
BACKUP VSAM_GROUP(name32)
{INVENTORY_ONLY(NO | YES)}
{PREFIX(YES | NO)}
OUTPUT_MODEL(name32
{LOCATION(locname8)
{COMMENT(comment_fields})
{STATUS( ACTIVE | INACTIVE )})
OUTPUT_MODEL(name32)
;}
```
To code a BACKUP VSAM_GROUP statement with the VSAM_NAME_MODEL keyword, use the following syntax:

```
BACKUP VSAM_GROUP(name32,*,...)
{INVENTORY_ONLY(NO | YES)}
{PREFIX(YES | NO)}
VSAM_NAME_MODEL(name32
{LOCATION(locname8)}
{COMMENT(comment_fields)}
{STATUS( ACTIVE | INACTIVE )} )
VSAM_NAME_MODEL(name32)
;
```

**INVENTORY_ONLY Keyword**

You can use the optional INVENTORY_ONLY keyword on the BACKUP VSAM_GROUP command to produce a report of the resources without actual execution of the backup. Because no execution occurs, no data sets are cataloged if you use this option.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

- **NO**  Produce a backup without a preliminary resource report.
- **YES**  Produce a preliminary report of VSAM and backup file resources, but prevent the backup process.

**OUTPUT_MODEL Keyword**

Use the OUTPUT_MODEL keyword to specify allocation information for backups of files in VSAM groups.
Using the RESTORE VSAM_GROUP Command

Use the RESTORE VSAM_GROUP command to rebuild VSAM clusters from a backup. You would perform a restore VSAM_GROUP when the current backup is the point or condition you want to establish for the VSAM group files.

**Syntax of RESTORE VSAM_GROUP**

The following restore command shows the new parameters introduced for group restore support. To code a RESTORE VSAM_GROUP statement, use the following syntax:

```plaintext
RESTORE {INVENTORY_ONLY(NO | YES)} VSAM_GROUP(name32) ;
```

**INVENTORY_ONLY Keyword**

You can use the optional INVENTORY_ONLY keyword on the RESTORE VSAM_GROUP command to check for desired results before actually performing a VSAM group restore. If YES is selected, full restore processing will not occur and a report will be produced. The INVENTORY_ONLY keyword does not simulate the complete performance of the command.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

- **YES** Produce a preliminary report of VSAM and backup file resources, but prevent the restore process.
- **NO** Produce a restore without a preliminary resource report.
START_TIME keyword

The START_TIME keyword is valid with the RESTORE VSAM_GROUP command. You can use this keyword with the STOP_TIME keyword to specify the backup data to restore.

Use the optional START_TIME keyword to specify the start (earliest) timestamp of the range of data to include in the restore process. This timestamp is required if you are using a backup that was not produced by RUV. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the START_TIME Keyword” on page 111.

STOP_TIME keyword

The STOP_TIME keyword is valid with the RESTORE VSAM_GROUP command. You can use this keyword with the START_TIME keyword to select the range of records to print from the archive.

Use the STOP_TIME keyword to specify the stop (latest) timestamp of the data to include in the restore process. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the RUV ISPF Interface” on page 113.
Using the RECOVER FORWARD VSAM_GROUP Command

You can use the forward recovery process to rebuild VSAM files or a VSAM group of files to a point in time, such as the completion of a particular job. The forward recovery process may use a backup of a file and the after-image journal records to rebuild the file.

Syntax of RECOVER FORWARD VSAM_GROUP

To code a RECOVER FORWARD VSAM_GROUP statement, use the following syntax:

```
RECOVER FORWARD
  {INVENTORY_ONLY(NO | YES)}
VSAM_GROUP(name32,*,...)
  BACKUP_IN(CURRENT_BACKUP)
  SELECTION_EXIT, (optional)
  START_TIME, (optional)
  STOP_TIME, (optional)
;
```

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the RECOVER FORWARD VSAM_GROUP command to check for desired results before actually performing a group forward recovery. If YES is selected, a report will be issued and the full forward recovery processing will not occur. The INVENTORY_ONLY keyword does not simulate the complete performance of the command.

Set one of the following values for the INVENTORY_ONLY keyword. The default is NO.

**YES**  Produce a preliminary report of VSAM and backup file resources, but prevent the forward recovery process.

**NO**   Produce a forward recovery without a preliminary resource report.
Example of a Group RECOVER FORWARD with New Names

The following example shows how to perform a forward recovery of a group and assign new names to the recovered data sets during the recovery. This task might be necessary at a disaster recovery site.

The example assumes that the data set names consist of four nodes. To adapt this example for your situation and build the correct VSAM_FILE_MASK value, you need to know how many nodes are used in your data set names.

The following command is executed first to create the VSAM name model for the group:

```sql
SET SUBSYSTEM(SSID)
REPORT(DETAIL);
ADD VSAM_NAME_MODEL(DRMODEL)
VSAM_FILE_MASK(HLQ1.HLQ2.HLQ3*,HLQ1.NEWHLQ2.HLQ3<NODE3(4)>.<NODE4>)
REPLACE(NO);
```

Then the following command is executed to perform the forward recovery of the group to the new-name files that are specified in the VSAM name model:

```sql
SET SUBSYSTEM(SSID)
REPORT(DETAIL);
RECOVER FORWARD
VSAM_GROUP(DRGROUP
START_TIME(20093400000000)
STOP_TIME(20093500000000)
VSAM_NAME_MODEL(DRMODEL));
```
Using the RECOVER UOW VSAM_GROUP Command

You can use the completed unit of work recovery process to rebuild VSAM files or a VSAM group of files to a point in time, such as the completion of a particular job. The UOW recovery process may use a backup of a file and the after-image journal records to rebuild the file. The UOW recovery process addresses in-flight transactions.

Syntax of RECOVER COMPLETED_UNIT_OF_WORK VSAM_GROUP

To code a RECOVER FORWARD VSAM_GROUP statement, use the following syntax:

```
RECOVER COMPLETED_UNIT_OF_WORK
   {INVENTORY_ONLY(NO | YES)}
VSAM_GROUP(name32,*,...)
   BACKUP_IN(CURRENT_BACKUP)
   SELECTION_EXIT, (optional)
   START_TIME, (optional)
;
```

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the RECOVER COMPLETED_UNIT_OF_WORK VSAM_GROUP command to check for desired results before actually performing a recovery. If YES is selected, a report will be issued and the recovery processing will not occur. The INVENTORY_ONLY keyword does not simulate the complete performance of the command.

Set one of the following values for the INVENTORY_ONLY keyword. The default is NO.

YES  Produce a preliminary report of VSAM and backup file resources, but prevent the recovery process.

NO   Produce a recovery without a preliminary resource report.
Using the RECOVER BACKOUT VSAM_GROUP Command

You can use RUV to back out (or undo) changes that have been made to VSAM files in a VSAM group. The backout recovery process does not use VSAM backup data sets. Instead, it uses the existing VSAM data set and applies before-image journal records to reverse the changes.

Syntax of RECOVER BACKOUT VSAM_GROUP

To code a RECOVER BACKOUT VSAM_GROUP statement, use the following syntax:

RECOVER BACKOUT
   {INVENTORY_ONLY(NO | YES)}
VSAM_GROUP(name32)
   START_TIME, (optional)
   STOP_TIME, (optional)
   ARCHIVE_IN (optional, repeatable)
;

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the RECOVER BACKOUT VSAM_GROUP command to check for desired results before actually performing a VSAM group backout recovery. If YES is selected, a report will be issued and full backout recovery processing will not occur. The INVENTORY_ONLY keyword does not simulate the complete performance of the command.

Set one of the following values for the INVENTORY_ONLY keyword. The default is NO.

YES Produce a preliminary report of VSAM and backup file resources, but prevent the backout recovery process.

NO Produce a backout recovery without a preliminary resource report.
Using the ARCHIVE ACCUM VSAM_GROUP Command

Use the ARCHIVE ACCUM command to gather VSAM groups into archives that can contain individual data sets and data set groups.

Use of the ARCHIVE ACCUM command is fully explained in Chapter 6, “Working with Archives.” The following material expands on that chapter to include the use of VSAM groups.

You can use the optional VSAM_GROUP keyword to identify the VSAM group names to be selected for a set of archive outputs. You can select the same VSAM group for more than one archive group. The VSAM_GROUP keyword is mutually exclusive with the UNSELECTED_VSAM_FILES keyword.

**NOTE**

If the VSAM_GROUP data set names are not found in the repository, or if the base of a specified path is not found, the accumulation process issues a condition code of 4.

The consolidated and last image approach to archive files and groups may be used at the same time.

**Syntax of ARCHIVE ACCUM VSAM_GROUP**

The following ARCHIVE command shows the new parameters introduced for group ARCHIVE support. To code an ARCHIVE ACCUM statement with VSAM_GROUP, use the following syntax:

```
ARCHIVE
  ACCUM
    { CONSOLIDATED
      { VSAM_FILE( ddn8_or_dsn44, *, ...)
        VSAM_GROUP( name32, *, ...)
        | UNSELECTED_VSAM_FILES
        { RETAIN_IMAGE(BOTH | LOG | JOURNAL) } } } |
    { LAST_IMAGE
      VSAM_FILE( ddn8_or_dsn44, *, ...)
      VSAM_GROUP( name32, *, ...)
      { RETAIN_IMAGE(JOURNAL) } }
  ARCHIVE_OUT(ddn8)
;`
```
CONSOLIDATED Keyword

You can use the CONSOLIDATED keyword to specify that RUV should write a consolidated archive with VSAM groups. The CONSOLIDATED keyword is the default.

RETAIN_IMAGE Keyword

You can use the optional RETAIN_IMAGE keyword to control which types of archive images to copy. BOTH is the default value. Specify one of the following values:

- BOTH
  Select before-images and after-images.

- LOG
  Select before-images.

- JOURNAL
  Select after-images.

---

**NOTE**

If the RETAIN_IMAGE keyword and LAST_IMAGE keyword are selected, JOURNAL is the only value that can be used with the RETAIN_IMAGE keyword.

---

LAST_IMAGE Keyword

Use the LAST_IMAGE keyword to specify that RUV select the most current journal images for forward recovery of VSAM data sets contained within a VSAM_GROUP. All records except the most current records are discarded.

---

**NOTE**

LAST_IMAGE data can be used only for forward recovery.
Using the ADD VSAM_SET Command

You can use VSAM groups to identify VSAM data sets for which you want RUV to record journal and log information. You can also use VSAM groups to specify the type of recovery that can be performed against those data sets. You generally group VSAM groups along with VSAM rules under one or more VSAM sets. Regardless of coding order, EXCLUDE files are built first, followed by INCLUDE files.

Syntax of ADD VSAM_SET

The following ADD VSAM_SET command shows the new parameters introduced for group VSAM_SET support. To code an ADD VSAM_SET statement, use the following syntax:

```
ADD VSAM_SET(name32)
COMMENT( comment_fields) 
VSAM_GROUP(name32
JOURNAL | LOG | BOTH )
VSAM_GROUP(name32)
VSAM_RULE(dsn_mask44,
EXCLUDE | LOG | JOURNAL | BOTH )
VSAM_RULE( ... )
; 
```
VSAM_GROUP Keyword

Use one or more VSAM_GROUP keywords on the ADD command to identify the VSAM data sets within a group and tell RUV what kind of recovery processing you want to enable. EXCLUDE is not a valid selection for VSAM_GROUP because the only excludes it honors are coded in the VSAM group definitions.

**NOTE**

RUV searches VSAM groups sequentially. Be cautious about the order in which you code them. RUV provides the TRY command as a mechanism for testing rules sets.

Use the following positional values on the VSAM_GROUP keyword:

**LOG**

- Use this value to cause RUV to record before-image records, providing backout capability for the file.

**JOURNAL**

- Use this value to cause RUV to record after-image records, providing forward-recovery capability for the file.

**BOTH**

- Use this value to cause RUV to record both before-images and after-images, providing backout and forward-recovery capability for the file.
Maintaining Groups

You maintain files in groups in exactly the way that you maintain individual VSAM files. Groups of VSAM files can be archived, designated for output to specific locations, updated, purged, deleted, and reported.

Using the ADD OUTPUT_MODEL Command

Use the ADD OUTPUT_MODEL command to provide allocation information for backups of files in VSAM groups.

Syntax of ADD OUTPUT_MODEL

The following ADD OUTPUT_MODEL command shows the new parameters introduced for group support. To code an ADD OUTPUT_MODEL statement, use the following syntax:

```
ADD_OUTPUT_MODEL(name32)
{ COMMENT(comment_fields) }
{ BACKUP_OUT_TYPE(DASD|TAPE|STACKED_TAPE) }
{ STACKED_TAPE_COUNT(nn) }
QSAM(" DSN=xxxx.xxxx.xxxx, "
" SPACE=(xxx,(xxx,xxx)), "
" UNIT=xxxxx ),DISP=(xxx) ")
;
```

BACKUP_OUT_TYPE Keyword

Use the optional BACKUP_OUT_TYPE keyword to identify the type of device where a group backup will be stored.

**DASD**

Use this value to specify DASD as the group backup output device. This value causes the STACKED_TAPE_COUNT keyword to be ignored.

**TAPE**

Use this value to specify tape as the group backup output device. RUV writes only one file to the tape.

**STACKED_TAPE**

Use this value to specify a stacked tape as the group backup output device. You can indicate the number of tapes that may be mounted with the STACKED_TAPE_COUNT keyword.
STACKED_TAPE_COUNT Keyword

Use the optional STACKED_TAPE_COUNT keyword to specify the number of group backup stacked tapes to be mounted, where nn specifies the number of tape devices. The default value is 1 per copy.

QSAM Keyword

You are responsible for ensuring that unique data set names are created for the output data sets. Several RUV variables can help you to do this. For more information about variables, see “Creating User Variables” on page 171 and “Using Variables” on page 173.

The required QSAM keyword provides allocation information that translates into the fields on a standard DD allocation statement. You provide the data set name and the same space and unit parameters that you would code on a typical z/OS DD statement. The only differences are that the //ddname DD portion is omitted and the data set name and other parameters may contain RUV or IBM substitution variables. Quotes are required where the values contain embedded commas.

QSAM allocations may contain any of the standard keywords and parameters listed in Table 23. For more information, see the corresponding JCL topics in the OS/390 documentation set.

RUV uses the same model definition for all VSAM files in the VSAM_GROUP. This presents no problem when backing up to tape drives but may cause problems when backing up to DASD. Provide a minimum primary space allocation to handle most files and a secondary space allocation to handle large files. Use the RLSE parameter to release unused DASD space.

Table 23 OUTPUT_MODEL Keywords and Parameters

<table>
<thead>
<tr>
<th>DATACLASS=</th>
<th>SPACE=</th>
<th>UNIT= (devname.,#)</th>
<th>DEFER</th>
<th>VOLUME= [or VOL=]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCB=</td>
<td>ABSTR</td>
<td>P</td>
<td>DEFER</td>
<td></td>
</tr>
<tr>
<td>BLKSIZE=</td>
<td>CYL</td>
<td>PRIVATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LRECL=</td>
<td>TRK</td>
<td>RETAIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSORG=</td>
<td>RLSE</td>
<td>SER=dsname</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECFM=</td>
<td>ALX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNAME= [or DSN=]</td>
<td>MXG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISP=</td>
<td>CONTIG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPDT= yyyyddd</td>
<td>ROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREE=</td>
<td>SYSDT= (class.,intrdr)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LABEL= (#,type)</td>
<td>STORCLAS=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MGMTCLASS=</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETPD= nnnn</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using the UPDATE OUTPUT_MODEL Command

Use the UPDATE OUTPUT_MODEL command to update allocation information for storage of VSAM groups.

Syntax of UPDATE OUTPUT_MODEL

The following UPDATE OUTPUT_MODEL command shows the new parameters introduced for group support. To code an UPDATE OUTPUT_MODEL statement, use the following syntax:

```
UPDATE OUTPUT_MODEL(name32)
   { COMMENT(comment_fields)  }
   { BACKUP_OUT_TYPE(DASD|TAPE|STACKED_TAPE) }
   { STACKED_TAPE_COUNT(nn) } 
   QSAM(" DSN=xxxx.xxxx.xxxx, ",
" SPACE=(xxx,(xxx,xxx)), "
" UNIT=xxxxx ),DISP=(xxx) ")
;                      
```

Using the DELETE OUTPUT_MODEL Command

Use the DELETE OUTPUT_MODEL command to delete allocation information for storage of VSAM groups.

Syntax of DELETE OUTPUT_MODEL

To code a DELETE OUTPUT_MODEL statement, use the following syntax:

```
DELETE OUTPUT_MODEL(name32);
```

Using the REPORT OUTPUT_MODEL Command

Use the REPORT OUTPUT_MODEL command to produce an OUTPUT_MODEL report.
Syntax of REPORT OUTPUT_MODEL

To code a REPORT OUTPUT_MODEL statement, use the following syntax:

```
REPORT OUTPUT_MODEL(name32,*,...):
```

Using the UPDATE VSAM_GROUP_DEFINITION Command

Use the UPDATE VSAM_GROUP_DEFINITION command to replace the VSAM_GROUP_DEFINITION with a new definition.

The UPDATE VSAM_GROUP_DEFINITION command combines the functions of delete and add into a single request.

Syntax of UPDATE VSAM_GROUP_DEFINITION

To code an UPDATE VSAM_GROUP_DEFINITION statement, use the following syntax:

```
UPDATE VSAM_GROUP_DEFINITION(name32)
{ COMMENT(comment_fields) }
VSAM_FILE(dsn44,
{ INCLUDE | EXCLUDE }
)
VSAM_FILE( ... )
;
```

Using the DELETE VSAM_GROUP_DEFINITION Command

Use the DELETE VSAM_GROUP_DEFINITION command to delete a VSAM_GROUP_DEFINITION name. Individual VSAM file registrations are not deleted.

Syntax of DELETE VSAM_GROUP_DEFINITION

To code a DELETE VSAM_GROUP_DEFINITION statement, use the following syntax:

```
DELETE VSAM_GROUP_DEFINITION(name32,*,...):
```
Using the DELETE VSAM_GROUP Command

Use the DELETE VSAM_GROUP command to delete the individual VSAM files that the group resolves to at run time. The definition is not deleted.

Syntax of DELETE VSAM_GROUP

To code a DELETE VSAM_GROUP statement, use the following syntax:

```
DELETE VSAM_GROUP(name32,*, ...) ;
```

Using the UPDATE VSAM_GROUP Command

Use the UPDATE VSAM_GROUP command to update information in the individual VSAM file repository records for the files that belong to the group.

Syntax of UPDATE VSAM_GROUP

To code an UPDATE VSAM_GROUP statement, use the following syntax:

```
UPDATE VSAM_GROUP(name32)
  { DAYS(nnn) }
  { CYCLES(nnn) }
  { SELECTION_EXIT }
;  
```

**DAYS Keyword**

Use the DAYS keyword to control the number of 24-hour days of historical data to retain in the repository.

The primary difference between using the DAYS and CYCLES keywords is that DAYS is an absolute number of 24-hour days, while CYCLES is a number of groups of data files (cycles) that contain a complete recoverable VSAM file.

Set the value of the DAYS keyword to the number (0 to 999) of days you want to retain.

If you code the DAYS keyword, do not code the CYCLES keyword.
**CYCLES Keyword**

Use the CYCLES keyword to control the number of iterations (cycles) of historical data to retain in the repository. Normally, this cycle includes the backup data and the archive data. The cycle begins when a backup is created for a VSAM file. All archive data created after this backup becomes part of this cycle. The next backup logically begins a new cycle.

Set the value of the CYCLES keyword to the number (0 to 999) of cycles you want to retain. RUV maintains the information about the cycles and their related data in the RUV repository. Setting a large number of cycles increases the number of data records in the repository and adds overhead to any processing that uses the repository. You should select a cycle number that protects your data and allows for any historical recovery requirements.

As new cycles are added to the repository, previous cycles are no longer needed. The data associated with old cycles remains in the repository until the PURGE REPOSITORY command is executed. This command removes the oldest cycle data from the repository until the number of cycles you have set with the CYCLES keyword remains.

If you code the CYCLES keyword, do not code the DAYS keyword.

**SELECTION_EXIT Keyword**

Use the SELECTION_EXIT keyword to have RUV invoke a user exit routine for each record that it processes. Specify the program name of the exit routine as the value of the SELECTION_EXIT keyword. For more information about using exit routines, see “Using the SELECTION_EXIT User Exit” on page 517.

**Using the REPORT VSAM_GROUP_DEFINITION Command**

Use the REPORT VSAM_GROUP_DEFINITION command to produce a VSAM_GROUP_DEFINITION report. To code a REPORT VSAM_GROUP_DEFINITION statement, use the following syntax:

```
REPORT VSAM_GROUP_DEFINITION(name32,*,...) ;
```
Using the REPORT VSAM_GROUP Command

Use the REPORT VSAM_GROUP command to produce a report on all files in the group. To code a REPORT VSAM_GROUP statement, use the following syntax:

```
REPORT VSAM_GROUP(name32,*,...) ;
```

Using the PURGE REPOSITORY VSAM_GROUP Command

Use the PURGE REPOSITORY command to remove information about registered backups and archives from the repository. See Chapter 14, “Maintaining the RUV Repository” for full use of the PURGE REPOSITORY command.

Syntax of PURGE REPOSITORY VSAM_GROUP

To code an optional PURGE REPOSITORY VSAM_GROUP statement, use the following syntax:

```
PURGE REPOSITORY
  { PURGE_OUT(ddn8) }
  { VSAM_FILE(*, ...) }
  { VSAM_GROUP(name32,*,...) }
  { SCRATCH( NO | YES ) }
  { SIMULATE } ;
```

PURGE REPOSITORY VSAM_GROUP Keyword

Use the PURGE REPOSITORY VSAM_GROUP keyword to restrict the purge operation to only those files within the specified VSAM_GROUP.
Chapter 16 Using Sequential Groups

This chapter introduces the Recovery Utility for VSAM (RUV) sequential group feature. The following topics are included:

Overview of Sequential Groups ................................................................. 624
  Group Definition ................................................................................. 624
  Group Use ......................................................................................... 624
  Group Backups ................................................................................. 624
  Group Restore .................................................................................... 625
  Group Maintenance ............................................................................ 625
  Group Reports ................................................................................... 625
  Group Purge ....................................................................................... 626
  Group Update ..................................................................................... 626
Defining Groups ...................................................................................... 627
  Using the ADD SEQ_GROUP_DEFINITION Command .......................... 627
  Coding the ADD SEQ_GROUP_DEFINITION Command ......................... 627
Using Groups .......................................................................................... 629
  Using the BACKUP SEQ_GROUP Command ....................................... 629
  Using the RESTORE SEQ_GROUP Command ...................................... 631
Maintaining Groups .................................................................................. 633
  Using the UPDATE SEQ_GROUP_DEFINITION Command .................... 633
  Using the DELETE SEQ_GROUP_DEFINITION Command ..................... 633
  Using the DELETE SEQ_GROUP Command ........................................ 634
  Using the UPDATE SEQ_GROUP Command ........................................ 634
  Using the REPORT SEQ_GROUP_DEFINITION Command .................... 635
  Using the REPORT SEQ_GROUP Command ........................................ 636
  Using the PURGE REPOSITORY SEQ_GROUP Command ...................... 636
Overview of Sequential Groups

The RUV sequential group feature provides a flexible method for working with a collection of RUV-registered sequential files. This is similar to the way that you work with a single file. After the sequential files have been registered and a sequential group has been defined, the SEQ_GROUP keyword can replace the SEQ_FILE keyword in most commands, including BACKUP, RESTORE, and REPORT.

When an RUV command is issued using sequential groups, the data sets contained in the group resolve to their sequential file(s) at run time as registered in the repository. Thus, if you specify sequential group definition using DSN masking, new sequential files will be selected automatically as soon as they are registered in the repository.

Group Definition

Group definitions are built by identifying data sets to be included or excluded. The sequential files can be specified by fully qualified DSNs, DSNs with masking, or a combination of both.

The group name is supplied by the user. The group name can have up to 32 characters and is stored in the RUV repository.

**NOTE**

A group definition without any INCLUDEs would be of no use for any command except rule sets. The order of the statements in the definition does not affect the results of any command.

Group Use

Once a sequential group is defined, RUV can perform backup and restore operations on multiple files. Group definitions allow the user to grow and change the number of sequential files managed without JCL changes.

Group Backups

RUV resolves and expands the group into sequential file names; this expansion is based on the sequential file names that are recorded in the RUV repository at execution time. The expanded group of files can be backed up in parallel to DASD, multiple tape drives, or stacked tapes, providing you with superior tape utilization.
For RUV to create RUV backups of a group of sequential files, you must use the OUTPUT_MODEL keyword (see page 630) to supply the data set names of the output files and their allocations. For RUV to create Instant Snapshot copies of a group of sequential files, you must use the VSAM_NAME_MODEL keyword to specify the name model to use for constructing the output data set names.

Backup data sets created by sequential group backups can be used for individual file restore, as well as sequential group restore and recovery.

**Group Restore**

Files in groups are restored in parallel provided that the memory resources and file resources are available.

Sequential group restore operations are not synchronized. The latest backup of each file is used, regardless of the source. For example, SMS backups can be used for some files, while RUV backups can be used for other files in the group. If you want to restore a data set from a backup that is not the current backup, you must use the individual sequential file restore command, where a BACKUP_IN can be named. Also, you can temporarily update the group definition to exclude the file, restore the group, and then add the file back into the group.

**Group Maintenance**

Sequential group maintenance applies to updating, purging, reporting, and deleting of sequential files by group. RUV does not allow group maintenance for ADD SEQ_FILE and REGISTER BACKUP_IN commands.

**Group Reports**

RUV reports can be used to verify the contents of each group. You can request a report that shows the following items:

- the group definition
- files that are in the group
- output environment for the groups

You can use the INVENTORY_ONLY(YES) parameter to display which files would be used for a particular command without actual processing of file content.
Group Purge

Sequential groups can be used to restrict the PURGE REPOSITORY command to work on only those backups and archives that contain data for files included in the sequential group.

Group Update

There are two versions of sequential group updates that are similar in appearance but have dramatically different results.

The first version updates the definition of a group. You can add, update, or delete sequential group definitions. You must provide a complete definition each time you add or update a group definition. Only the definition is changed; individual files are not affected.

The second version updates information for the individual sequential files that the sequential group resolves to at run time. This provides a convenient method to add or alter parameters of all sequential files in the group. For example, you can add or change the number of days or cycles that control purge processing for a group of sequential files.
Defining Groups

You define sequential groups by using the ADD SEQ_GROUP_DEFINITION command of the RUVZSM0 utility.

Using the ADD SEQ_GROUP_DEFINITION Command

Use the ADD SEQ_GROUP_DEFINITION command to create a sequential group record in the repository.

This group can have a name of up to 32 characters. It can have a comment, but must have at least one sequential file or DSN mask defining which file(s) with which to work. Each DSN mask can be fully qualified or have wildcards. The mask(s) will resolve to individual sequential files at each run time using only files registered in the repository.

Coding the ADD SEQ_GROUP_DEFINITION Command

To code an ADD SEQ_GROUP_DEFINITION statement, use the following syntax:

```
ADD SEQ_GROUP_DEFINITION(name32)
     { COMMENT(comment_fields) }  
SEQ_FILE(dsn44,
     { INCLUDE | EXCLUDE }
     )  
SEQ_FILE(...)
;
```

COMMENT Keyword

The COMMENT keyword is available for use with RUV sequential group commands and keywords that describe RUV sequential groups. You can use it for documentation or other purposes to help you control recovery information.

Code the COMMENT keyword as shown in the following example. For the comment value, you can code a maximum of 200 bytes in four 50-byte fields. Enclose each field in quotes if the field contains any spaces. Enclose all fields in parentheses.

```
COMMENT('this is a comment'  
 'this is more information'  
 'this is a third line of information'  
 'this is the last line of the comment')
```
**SEQ_FILE Keyword**

Use the required SEQ_FILE keyword to identify the sequential data set names to be selected or excluded as members of SEQ_GROUP_DEFINITION. The SEQ_FILE can have masking or be fully qualified data set names. You can select the same sequential file for more than one sequential group. Each SEQ_GROUP is completely independent of all other SEQ_GROUP.

EXCLUDE

Use the optional EXCLUDE value to prevent a particular sequential data set from being selected into a sequential group. For example, if you have a list of sequential data sets that were selected by using a wildcard, you might want to specify one or more data sets in that list for exclusion from the group.

INCLUDE

Use the optional INCLUDE value to select one or more particular sequential data sets in a group.

---

**NOTE**

EXCLUDE and INCLUDE are mutually exclusive values. During RUV processing of groups, the EXCLUDE value has precedence over the INCLUDE value.
Using_groups

Sequential groups can be used in combination with individual files, depending on your data recovery needs. RUV can use sequential groups for backup, restore, recovery, and archive operations.

RUV automatically registers backups and archives in the repository as they are created, and this applies to those created from sequential groups. While this information is required to successfully build recovery control job streams, this information needs to be deleted from the repository when the group data is no longer needed. You use the PURGE command to perform this function.

See Chapter 14, “Maintaining the RUV Repository” for more information on the PURGE REPOSITORY command.

Using the BACKUP SEQ_GROUP Command

Use the BACKUP SEQ_GROUP command to back up multiple sequential files with a single command. Files will be backed up in parallel provided that memory is sufficient and that the OUTPUT_MODEL keyword provides multiple output devices.

Syntax of BACKUP SEQ_GROUP

The following backup command shows the new parameters introduced for group backup support. To code a BACKUP SEQ_GROUP statement with the OUTPUT_MODEL keyword, use the following syntax:

```
BACKUP SEQ_GROUP(name32)  
{INVENTORY_ONLY(NO | YES))
{PREFIX(YES | NO))
OUTPUT_MODEL(name32 
{LOCATION(locname8)
{COMMENT(comment_fields})
{STATUS( ACTIVE | INACTIVE ))} )
OUTPUT_MODEL(name32)
;
```
To code a BACKUP SEQ_GROUP statement with the VSAM_NAME_MODEL keyword, use the following syntax:

```
BACKUP SEQ_GROUP(name32,*,...)  
 {INVENTORY_ONLY(NO | YES)}  
 {PREFIX(YES | NO)}  
 VSAM_NAME_MODEL(name32)  
 {LOCATION(locname8)  
 {COMMENT(comment_fields)  
 {STATUS( ACTIVE | INACTIVE )} }}  
 VSAM_NAME_MODEL(name32)  
 ;
```

**INVENTORY_ONLY Keyword**

You can use the optional INVENTORY_ONLY keyword on the BACKUP SEQ_GROUP command to produce a report of the resources without actual execution of the backup. Because no execution occurs, no data sets are cataloged if you use this option.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

- **NO**  Produce a backup without a preliminary resource report.
- **YES**  Produce a preliminary report of sequential and backup file resources, but prevent the backup process.

**OUTPUT_MODEL Keyword**

Use the OUTPUT_MODEL keyword to specify allocation information for backups of files in sequential groups.
Using the RESTORE SEQ_GROUP Command

You would perform a restore SEQ_GROUP when the current backup is the point or condition you want to establish for the sequential group files.

Syntax of RESTORE SEQ_GROUP

The following restore command shows the new parameters introduced for group restore support. To code a RESTORE SEQ_GROUP statement, use the following syntax:

```
RESTORE
  {INVENTORY_ONLY(NO | YES)}
SEQ_GROUP(name32)
;
```

INVENTORY_ONLY Keyword

You can use the optional INVENTORY_ONLY keyword on the RESTORE SEQ_GROUP command to check for desired results before actually performing a sequential group restore. If YES is selected, full restore processing will not occur and a report will be produced. The INVENTORY_ONLY keyword does not simulate the complete performance of the command.

Set one of the following values for the INVENTORY_ONLY keyword. The default value is NO.

**YES**  Produce a preliminary report of sequential and backup file resources, but prevent the restore process.

**NO**   Produce a restore without a preliminary resource report.
**START_TIME keyword**

The START_TIME keyword is valid with the RESTORE SEQ_GROUP command. You can use this keyword with the STOP_TIME keyword to specify the backup data to restore.

Use the optional START_TIME keyword to specify the start (earliest) timestamp of the range of data to include in the restore process. This timestamp is required if you are using a backup that was not produced by RUV. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the START_TIME Keyword” on page 111.

**STOP_TIME keyword**

The STOP_TIME keyword is valid with the RESTORE SEQ_GROUP command. You can use this keyword with the START_TIME keyword to select the range of records to print from the archive.

Use the STOP_TIME keyword to specify the stop (latest) timestamp of the data to include in the restore process. The timestamp could be required for a point-in-time restore, depending on the restore situation.

For more information, see “Using the RUV ISPF Interface” on page 113.
Maintaining Groups

You maintain files in groups in exactly the way that you maintain individual sequential files. Groups of sequential files can be updated, purged, deleted, and reported.

Using the UPDATE SEQ_GROUP_DEFINITION Command

Use the UPDATE SEQ_GROUP_DEFINITION command to replace the SEQ_GROUP_DEFINITION with a new definition.

The UPDATE SEQ_GROUP_DEFINITION command combines the functions of delete and add into a single request.

Syntax of UPDATE SEQ_GROUP_DEFINITION

To code an UPDATE SEQ_GROUP_DEFINITION statement, use the following syntax:

```
UPDATE SEQ_GROUP_DEFINITION(name32)
   { COMMENT(comment_fields) }
SEQ_FILE(dsn44,
   { INCLUDE | EXCLUDE }
   )
SEQ_FILE( ... )
;
```

Using the DELETE SEQ_GROUP_DEFINITION Command

Use the DELETE SEQ_GROUP_DEFINITION command to delete a SEQ_GROUP_DEFINITION name. Individual sequential file registrations are not deleted.

Syntax of DELETE SEQ_GROUP_DEFINITION

To code a DELETE SEQ_GROUP_DEFINITION statement, use the following syntax:

```
DELETE SEQ_GROUP_DEFINITION(name32,*, ...) ;
```
Using the DELETE SEQ_GROUP Command

Use the DELETE SEQ_GROUP command to delete the individual sequential files that the group resolves to at run time. The definition is not deleted.

Syntax of DELETE SEQ_GROUP

To code a DELETE SEQ_GROUP statement, use the following syntax:

```
DELETE SEQ_GROUP(name32,*, ...) ;
```

Using the UPDATE SEQ_GROUP Command

Use the UPDATE SEQ_GROUP command to update information in the individual sequential file repository records for the files that belong to the group.

Syntax of UPDATE SEQ_GROUP

To code an UPDATE SEQ_GROUP statement, use the following syntax:

```
UPDATE SEQ_GROUP(name32)
{ DAYS(nnn) }
{ CYCLES(nnn) }
{ SELECTION_EXIT }
;
```

DAYS Keyword

Use the DAYS keyword to control the number of 24-hour days of historical data to retain in the repository.

The primary difference between using the DAYS and CYCLES keywords is that DAYS is an absolute number of 24-hour days, while CYCLES is a number of groups of data files (cycles) that contain a complete recoverable sequential file.

Set the value of the DAYS keyword to the number (0 to 999) of days you want to retain.

If you code the DAYS keyword, do not code the CYCLES keyword.
CYCLES Keyword

Use the CYCLES keyword to control the number of iterations (cycles) of historical data to retain in the repository. Normally, this cycle includes the backup data and the archive data. The cycle begins when a backup is created for a sequential file. All archive data created after this backup becomes part of this cycle. The next backup logically begins a new cycle.

Set the value of the CYCLES keyword to the number (0 to 999) of cycles you want to retain. RUV maintains the information about the cycles and their related data in the RUV repository. Setting a large number of cycles increases the number of data records in the repository and adds overhead to any processing that uses the repository. You should select a cycle number that protects your data and allows for any historical recovery requirements.

As new cycles are added to the repository, previous cycles are no longer needed. The data associated with old cycles remains in the repository until the PURGE REPOSITORY command is executed. This command removes the oldest cycle data from the repository until the number of cycles you have set with the CYCLES keyword remains.

If you code the CYCLES keyword, do not code the DAYS keyword.

SELECTION_EXIT Keyword

Use the SELECTION_EXIT keyword to have RUV invoke a user exit routine for each record that it processes. Specify the program name of the exit routine as the value of the SELECTION_EXIT keyword. For more information about using exit routines, see “Using the SELECTION_EXIT User Exit” on page 517.

Using the REPORT SEQ_GROUP_DEFINITION Command

Use the REPORT SEQ_GROUP_DEFINITION command to produce a SEQ_GROUP_DEFINITION report. To code a REPORT SEQ_GROUP_DEFINITION statement, use the following syntax:

```
REPORT SEQ_GROUP_DEFINITION(name32,*,*,...) ;
```

Using the REPORT SEQ_GROUP Command

Use the REPORT SEQ_GROUP command to produce a report on all files in the group. To code a REPORT SEQ_GROUP statement, use the following syntax:

```
REPORT SEQ_GROUP(name32,*,...) ;
```

Using the PURGE REPOSITORY SEQ_GROUP Command

Use the PURGE REPOSITORY command to remove information about registered backups and archives from the repository. See Chapter 14, “Maintaining the RUV Repository” for full use of the PURGE REPOSITORY command.

Syntax of PURGE REPOSITORY SEQ_GROUP

To code an optional PURGE REPOSITORY SEQ_GROUP statement, use the following syntax:

```
PURGE REPOSITORY
 { PURGE_OUT(ddn8) } 
 { SEQ_FILE(*, ...) } 
 { SEQ_GROUP(name32,*,...) } 
 { SCRATCH(NO | YES) } 
 { SIMULATE } 
 ;
```

PURGE REPOSITORY SEQ_GROUP Keyword

Use the PURGE REPOSITORY SEQ_GROUP keyword to restrict the purge operation to only those files within the specified SEQ_GROUP.
Solving Problems

This appendix provides information you can use if a problem occurs with RUV. The following information is included:

Collecting Problem Information ..................................................... 638
Diagnostic Steps ........................................................................... 639
Displaying PTF Details ................................................................. 641
Displaying Environment Information with the RUVZSM0 Utility ........... 642
Displaying Environment Information with the RUV ISPF Interface .......... 643
Collecting Problem Information

For any problems, you should first produce reports that will display the current state of the repository.

After collecting the reports, you should review the contents. It may be necessary to run a report (of some report type) before a job is run and then again after the job completes to verify the activity that is occurring. You need the following types of documentation:

1. Produce the following documentation in the order given:
   - job log of the failing action
   - console log of the failing job
   - SYSMDUMP as necessary
   - description of what is failing
   - list recent changes to the system
   - OS level
   - CPU type
   - Other vendor compression products (VIO Plus, IAM)

2. Review and react to any messages that occurred, starting with the earliest message in the job

3. Run appropriate reports, before and after the failure.

4. Use SDATA=(RGN, LSQA, LPA, NUC, SQA, CSA, TRT) if you produce a SLIP dump or address space dump “DUMP COMM=(comments)”.

If you encounter a problem using RUV and call Customer Support, you might be asked to perform one or more diagnostic steps.
Diagnostic Steps

The following diagnostic steps produce information that help you and Product Support understand the problem.

1 Run the VSAM Trace facility. You must add the following AMP JCL subparameter to the appropriate DD statement:

```
//ddname DD DSN=...<cluster name>...,DISP=SHR,
   AMP='TRACE'
```

To start the VSAM Trace facility and print the trace, perform the following steps:

A Start the generalized trace facility (GTF) with the TRACE=USR option.

B You can print the trace by executing the GTXTRACE subcommand under IPCS.

2 Run the IBM Enhanced VSAM Record Management Trace (RMTRACE).

To start RMTRACE and print the trace, perform the following steps:

C To invoke the VSAM RMTRACE, you must add the following AMP JCL subparameters to the appropriate DD statement. This statement also invokes the RPCV Trace facility.

```
//ddname DD DSN=...<cluster name>...,DISP=SHR,
   AMP='TRACE=(PARM1=400001811F20,
                      HOOK=(0,1,17))',
   SDATA=(RGN,LSQA,LPA,NUC,SQA,CSA,TRT)
```

D Start GTF with the TRACE=USR option.

E You can print the trace by executing the GTXTRACE subcommand under IPCS.

3 Take a SLIP DUMP and specify the following SDATA parameters:

```
SDATA=(RGN,LSQA,LPA,NUC,SQA,CSA,TRT)
```

Take an address space DUMP. Enter the following command at the z/OS console:

```
DUMP COMM=(comment)
```
4 Reply to the operator message specifying the following command:

```
R xx,ASID=xxxx,SDATA=(RGN,LSQA,LPA,NUC,SQA,CSA,TRT)
```

**NOTE**

You can also perform the following actions:

Add the following DD statement:

```
//RUVTRACE DD SYSOUT=*  
```

Then add the following SET statements:

```
SET SUBSYS(ssid)
SET REPORT(DETAIL)
...
TRACE_AREA_ON(put);
```
Displaying PTF Details

You can obtain information about the compile date and time of all RUV programs as well as any PTFs that have been applied to the system. To do so, use the REPORT PTF command of the RUVZSM0 utility.

PTF information typically is needed only when requested by Product Support. It can reveal load library problems (such as may be caused by mixing modules from different RUV releases in a single library).

To obtain PTF information in batch, execute the RUVZSM0 utility with the REPORT PTF command. The resulting PTF report (Figure 156) displays the compile date and time of all RUV programs and lists any PTFs that have been applied to the system. Data for the PTF display can come from the load library pointed to by a //SYSUT1 DD statement or the current STEPLIB. You can execute a PTF report against modules that have been loaded by a subsystem by specifying the SUBSYSTEM keyword on the REPORT command.

The resulting report is shown in Figure 157.

Figure 156  PTF Full Report

<table>
<thead>
<tr>
<th>Program</th>
<th>PID</th>
<th>Csect</th>
<th>Date</th>
<th>Time</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUVAACM</td>
<td>RUVACCM</td>
<td>01SEP07</td>
<td>17:13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVC99</td>
<td>10AUG07</td>
<td>15:13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUVZSCR</td>
<td>10AUG07</td>
<td>15:29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBTASK</td>
<td>10AUG07</td>
<td>15:03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZTRACE</td>
<td>10AUG07</td>
<td>15:32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUEUE</td>
<td>10AUG07</td>
<td>14:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMA</td>
<td>10AUG07</td>
<td>13:53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSGS</td>
<td>10AUG07</td>
<td>14:26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRINTER</td>
<td>10AUG07</td>
<td>14:41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIST</td>
<td>10AUG07</td>
<td>14:19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITEM</td>
<td>10AUG07</td>
<td>14:11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZTOD</td>
<td>10AUG07</td>
<td>15:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARCHREG</td>
<td>10AUG07</td>
<td>13:56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQUASH</td>
<td>10AUG07</td>
<td>15:07</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUV</td>
<td>RUXXO5D@ 08/10/2009</td>
<td>17:17</td>
<td>01:02:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUV</td>
<td>RUXXLJB@ 08/10/2009</td>
<td>17:15</td>
<td>01:02:00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Displaying Environment Information with the RUVZSM0 Utility

You can display RUV version and environment information by adding the /V parameter to the EXEC statement of the RUVZSM0 utility:

```plaintext
// EXEC PGM=RUVZSM0,PARM='/V'
```

### Figure 157 Information Produced with /V Parameter

<table>
<thead>
<tr>
<th>Compiled: Sep 27 2006 16:45:48</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVS/SP-MVS/SP 06.00.04</td>
</tr>
<tr>
<td>SP6.0.4 JBB6604 03B</td>
</tr>
<tr>
<td>SMFID: SYSP</td>
</tr>
<tr>
<td>SAF: 1.8</td>
</tr>
<tr>
<td>HSM: active</td>
</tr>
<tr>
<td>SMS: 1.4.0</td>
</tr>
<tr>
<td>Userid: ISIRAM</td>
</tr>
<tr>
<td>DDN    DSN</td>
</tr>
<tr>
<td>STEPLIB RUVRUVV1.SCLMRAM.LOAD</td>
</tr>
<tr>
<td>RUVRUVV1.SCLM1.LOAD</td>
</tr>
<tr>
<td>RUVRUVV1.SCLM2.LOAD</td>
</tr>
<tr>
<td>RUVRUVV1.SCLM3.LOAD</td>
</tr>
<tr>
<td>SYSPRINT ISIRAM.ISIRAMRR.JOB09457.D0000102.?</td>
</tr>
<tr>
<td>RUVTRACE ISIRAM.ISIRAMRR.JOB09457.D0000103.?</td>
</tr>
<tr>
<td>DUM    NULLFILE</td>
</tr>
<tr>
<td>SYSUDUMP ISIRAM.ISIRAMRR.JOB09457.D0000104.?</td>
</tr>
<tr>
<td>SYSPRIN ISIRAM.ISIRAMRR.JOB09457.D0000101.?</td>
</tr>
</tbody>
</table>
Displaying Environment Information with the RUV ISPF Interface

You can display additional RUV version and environment information by selecting the Help action bar choice and then selecting the About system choice (system is the z/OS system ID). See Figure 158.

Figure 158  About System Panel

<table>
<thead>
<tr>
<th>File</th>
<th>View</th>
<th>Tools</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>Recovery Utility for VSAM</td>
<td></td>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>RUV Options</td>
<td>2009.272 17:30</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>Subsystem ID . .</td>
<td>About SYSP</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>1. Perform a</td>
<td>SYSPLEX ID . . . . : LOCAL</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>2. VSAM DSN</td>
<td>Operating System : SP6.0.4</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>3. Applid /</td>
<td>RACF Version . . . : V2.04.0</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>4. Archive D</td>
<td>RACF Status . . . : AVAILABLE</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>5. Backup Fi</td>
<td>DFP Version . . . : V1.04.0</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>6. Log of Lo</td>
<td>DFHSM Version . . . : V1.04.0</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>7.Subsystem</td>
<td>SMS Status . . . . : ACTIVE</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td>8. Message L</td>
<td>TSO Version . . . . : V2.06.0</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>RUV Start</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>Parameters</td>
<td></td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>F1=Help</td>
<td>F10=Resize</td>
<td>F12=Cancel</td>
</tr>
<tr>
<td></td>
<td>F1=Help</td>
<td>F3=Exit</td>
<td>F10=Actions</td>
</tr>
</tbody>
</table>

Command ===> '-----------------------------------------'
F1=Help      F3=Exit     F10=Actions  F12=Cancel
Migrating from RPCV to RUV

If you are migrating from the BMC Software RECOVERY PLUS for CICS/VSAM (RPCV™) product to the Recovery Utility for VSAM (RUV) product, this appendix provides information to help you plan and perform the migration. The following topics are included:

Overview of Migration from RPCV to RUV ........................................... 646
Using the RUVCJC Utility ................................................................. 647
Using the RUVCDA Utility ............................................................... 649
   Creating Data Set, Group Definition, and User Variable Statements .... 650
   Defining Data Sets, Group Definitions, and User Variables to RUV .... 652
Identifying Jobs That Use RPCV .................................................. 653
RPCV-to-RUV Migration Considerations ........................................ 654
   Supported File Structures ............................................................ 654
   Repository Information for VSAM File Recovery ......................... 654
   System Requirements ................................................................. 654
   Rule Sets ................................................................................. 655
   Security ..................................................................................... 655
   ISPF Interface ........................................................................... 656
Using BMC Software Subsystems .................................................. 656
Batch Journaling and Job Step Logging ......................................... 656
Backing Up Master Data Sets ......................................................... 657
Purge Processing ........................................................................ 658
Archiving with RUV ................................................................... 658
RUV Recovery ............................................................................ 658
Creating RUV Reports ................................................................. 659
Miscellaneous Items .................................................................. 659
Overview of Migration from RPCV to RUV

RUV provides the following utilities to help you with migration from RPCV to RUV:

- RUVZCJC migrates JCT/OPT tables to RUV rule sets.
- RUVZCDA migrates VSAM data set name, group definitions, and user variables from the ISICDS data sets.

In addition, you can use the IBM ISRSUPC utility to search your JCL libraries and identify jobs that execute RPCV steps. You can then review these steps to determine how to handle them during the migration to RUV.

As you migrate from RPCV to RUV, you should keep in mind the considerations that are described in “RPCV-to-RUV Migration Considerations” on page 654.
Using the RUVZCJC Utility

RPCV currently requires you to assemble an ISIOPT and a VSAM DSN selection and ISIJCT. The ISIOPT is loaded into the common storage area (CSA) and used to provide a fast path reject of RPCV interest in a VSAM file. The ISIJCT is loaded into the address space when a VSAM file is opened. The ISIJCT provides job/step/program filters by DSN as well as journaling needs (before-images and/or after-images) and journal allocation specifications. The manual process is cumbersome and error prone.

RUVZCJC is a ISIJCT/ISIOPT migration utility that provides templates and models for converting your existing ISIJCT load module to RUV rule sets. RUVZCJC provides approximately 80 percent of your rule sets from the information in their existing ISIJCT load module.

**NOTE**

Because of fundamental differences between the way that RPCV and RUV view journaling (one journal per VSAM cluster and one journal per JOB step, respectively), RUVZCJC cannot provide 100 percent of the information you need for RUV from the information in the ISIJCT alone. After the migration is complete, the resulting rule set may require modification for accurate results.

Figure 159 shows sample JCL for executing RUVZCJC. The source for this sample is located in member CONVJCT of the RUV.CNTL library. The ISIJCT load module is located in the SYSLIB DD statement. The new RUV rule set is placed in the CARDS DD statement.

**Figure 159  Sample JCL for RUVZCJC**

```plaintext
//   . . . jobcard . . .
// *
//CONVJCT  EXEC PGM=RUVZCJC
//STEPLIB  DD DISP=SHR,DSN=yourname.LOAD <= RUV loadlib
//SYSLIB   DD DISP=SHR,DSN=your.RPCV.load <= Loadlib with ISIJCT
//CARDS    DD DSN=your.output.ruleset.name,<= New Rule Set
//          SPACE=(TRK,(5,5),RLSE),
//          DISP=(NEW,CATLG),
//          UNIT=SYSDA
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT2   DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT5   DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSPRINT DD SYSOUT=*<
//SYSSIN   DD *
ISIJCT
/*
```
Executing RUVZCJC

To use RUVZCJC, perform the following steps:

1. Review the JCL to verify that the UNIT parameters match your system requirements.

2. Modify the JOB statement to meet your system requirements.

3. Point the STEPLIB DD to the load library where you installed RUV.

4. Point the SYSLIB DD to the load library where your current ISIJCT load module is located.

5. Modify the PARM statement to include `PARM='!LMOD=ISIJCT'`, or specify the SYSIN DD statement as

   ```
   //SYSIN DD *
   ISIJCT
   /*
   ```

6. Point the CARDS DD statement to a data set that is defined with `RECFM=FB` and `LRECL=80`.

   The data set is either a PDS member

   ```
   //CARDS DD DISP=SHR,DSN=yourname(RUVCARDS)
   ```

   or a sequential data set

   ```
   //CARDS DD DISP=SHR,DSN=yourname.sequential.dataset
   ```

7. Submit the job to convert the existing CICS journal definitions to the new RUV rule set format.

8. Review the CARDS output, and modify the rule set to meet your site standards.

9. Check and complete building the entire rule set.

10. Activate the rule set as described in “Activating a Rule Set” on page 198.

11. You can use the journals created by RPCV in RUV if you use the ARCHIVE RPCV_IN command to convert them to a format that RUV can use. For more information, see “Archiving RPCV Journals” on page 272.
Using the RUVZCDA Utility

RUVZCDA converts RPCV data set name, group definitions, and user variables to produce RUV control statements. In addition to user variables, a set of system variables is included also. All unique RPCV data set names that are contained in the ISICDS are converted to RUV ADD VSAM_FILE statements. All RPCV groups are converted to RUV ADD VSAM_GROUP_DEFINITION statements. All RPCV user variables are converted to RUV ADD USER_VARIABLES statements.

None of the following parameters associated with ADD VSAM_FILE are migrated from RPCV:

- COMMENT
- SELECTION_EXIT
- DAYS

Both of the following VSAM_GROUP_DEFINITION parameters are converted from RPCV:

- VSAM_FILE
- COMMENT

However, COMMENT is included only when it exists in RPCV.

User variables are given a name indicating the applid, fileid, and/or group they are associated with.

**NOTE**

Because of fundamental differences between RPCV and RUV, RUVZCDA cannot provide 100 percent of the information that you need for RUV from the information in RPCV alone. After RUVZCDA is executed, the resulting RUV statements require modification for accurate results.

Using RUVZCDA is a two-step process. First you execute RUVZCDA to create RUV data set, group definition, and user variable statements. Then you use these statements to associate the data sets, group definitions, and user variables to RUV.
Creating Data Set, Group Definition, and User Variable Statements

Figure 160 shows sample JCL for executing RUVZCDA. The source for this sample is located in member CONVDSN1 of the RUV.CNTL library. The utility writes the new RUV data set, group definition, and user variable statements to the SYSPUNCH data set.

Figure 160  Sample JCL for RUVZCDA

```
//   . . . jobcard . . .
//*
//* -----------------------------------------------------
//* NOTE: See RUV User Guide Appendix B for instructions.
//* -----------------------------------------------------
//*
// CONVDSNS EXEC PGM=RUVZCDA
// STEPLIB DD DISP=SHR,DSN='yourname'.LOAD <= RUV Loadlib
// DD DISP=SHR,DSN='yourname'.LOAD <= RPCV Loadlib
// ISICDS DD DISP=SHR,DSN=your.RPCV.ISICDS <= RPCV CDS pair
// ISICDS1 DD DISP=SHR,DSN=your.RPCV.ISICDS1 <= RPCV CDS pair
// SYSPUNCH DD DSN=your.output.dsns.groups.name, <= RUV DSNs/GRPs stmts
// SPACE=(TRK,(5,5),RLSE),
// DD DISP=(NEW,CATLG),UNIT=SYSDA
// SYSPRINT DD SYSP=*
// CRDPRINT DD SYSP=*
/*
```

Reading the ISICDS

You can specify the ISICDS in either of the following ways:

- The easier way is to code the ISICDS DD statement and, if necessary, the ISICDS1 DD statement in the utility JCL. (Older versions of RPCV use one ISICDS, and ISICDS1 is not necessary.)

- Alternatively, you can allow the utility to use the ISICDSs coded in the ISICOPT by omitting the ISICDS and ISICDS1 DD statements from the utility JCL. If you use this approach, ISICOPT must be present in the STEPLIB.

Coding either ISICDS or ISICDS1 as DD DUMMY results in an abend.

For RUV to be able to read the RPCV ISICDS data set, the RPCV load lib must be present in the STEPLIB concatenation.
Executing RUVZCDA

To execute RUVZCDA, perform the following steps:

1. Review the JCL to verify that the `UNIT` parameters match your system requirements.

2. Modify the JOB statement to meet your system requirements.

3. Point the first DD in the STEPLIB concatenation to the load library where you installed RUV. Point the second DD in the STEPLIB concatenation to your RPCV library that contains the RPCV load lib.

4. Point the ISICDS DD and ISICDS1 DD statements to your primary and alternate CDS libraries.

   **NOTE**

   You can omit these statements if they are coded in the ISICOPT.

5. Point the SYSPUNCH DD statement to a data set that is defined with `RECFM=FB` and `LRECL=80`.

   The data set is either a PDS member

   ```
   //SYSPUNCH DD DISP=SHR,DSN='yourname(RUVDSNG)'
   ```

   or a sequential data set

   ```
   //SYSPUNCH DD DISP=SHR,DSN='yourname.sequential.dataset'
   ```

6. Submit the job. The utility writes RUV data set name and group definition statements to SYSPUNCH.
Defining Data Sets, Group Definitions, and User Variables to RUV

After you execute RUVZCDA, you are ready to use the control statements produced by the utility to define the data sets, group definitions, and user variables to RUV. Use the sample JCL shown in Figure 161. The source for this sample is located in member CONVDSN2 of the RUV.CNTL library.

Figure 161  Sample RUV Data Set Name and Group Definition and Report JCL

//   . . . jobcard . . .
//JOBLIB   DD DISP=SHR,DSN='yourname'.LOAD <= RUV Loadlib
//*
//DEFINE   EXEC PGM=RUVZSM0,REGION=64M
//SYSPRINT DD SYSOUT=*  <= From CONVDSN1
//SYSIN    DD DSN=your.output.dsns.groups.name, Disp=SHR
//*
//REPORT   EXEC PGM=RUVZSM0,REGION=64M
//SYSPRINT DD SYSOUT=*  <= From CONVDSN1
//SYSIN DD *
//SET SUBSYSTEM(ssid);
//SET REPORT(DETAIL);
//REPORT VSAM_FILE(*);
//REPORT VSAM_GROUP_DEFINITION(*);

Defining Data Sets and Groups to RUV

To define data sets and groups to RUV, perform the following steps:

1  Review the SYSPRINT report and the SYSPUNCH output from RUVZCDA. Modify the statements to meet your site requirements.

2  Use the sample JCL in Figure 161 as a model for the job to generate the RUV data set name and group definitions.

3  Modify the JOBLIB to meet your system requirements.

4  Point the first SYSIN in the job to the data set that contains your modified RUV statements.

5  Change ssid to your site RUV subsystem ID.

6  Submit the job.

7  Review the SYSPRINT of each step, and modify the RUV statements as needed. Repeat this step until all definitions have been generated correctly.
Identifying Jobs That Use RPCV

During the migration from RPCV to RUV, you should examine existing jobs that execute RPCV steps and determine how to handle them. You might need to replace some RPCV job steps with similar RUV job steps. You might be able to replace many RPCV job steps with a single RUV job step because RUV provides group processing that simplifies processes. You might be able to delete some RPCV job steps altogether because RUV provides superior automation.

To identify jobs that contain RPCV job steps, you can use a search utility, such as the IBM ISRSUPC utility, to search your JCL libraries. A sample job for performing this search is located in member ISRSRCH of the RUV.CNTL library. Figure 162 shows JCL to perform the search.

Figure 162  ISRSUPC JCL to Search JCL Library

```
//ISRSRCH JOB your job statement
//SEARCH EXEC PGM=ISRSUPC,PARM=('SRCHCMP,ANYC')
//NEWDD DD DSN=your.JCLLIB,DISP=SHR
//OUTDD DD SYSOUT=*
//SYSIN DD *
SRCHFOR 'ISIDRS'
  SRCHFOR 'ISIDRS',W,10:20
/*
```
RPCV-to-RUV Migration Considerations

The following sections describe RPCV-to-RUV migration considerations.

Supported File Structures

**Purpose:** RUV supports and recovers the following file structures and organizations: KSDS, ESDS, RRDS, VRRDS, and QSAM.

**New:** None

**Changed:** None

**Obsolete:** BDAM file type is no longer supported.

Repository Information for VSAM File Recovery

**Purpose:** The repository records information on data sets, associated journal records, and backups. The information is used by the RUV automatic recovery facility.

**New:** A new repository (and a REGISET2 repository) are now required. The following repository administrative commands are available: BACKUP REPOSITORY, RESTORE REPOSITORY, and PURGE REPOSITORY

**Changed:** None

**Obsolete:** Migration program RUVZCDA converts data set names and group definitions from the CDS to the new RUV product repository.

System Requirements

**Purpose:** RUV requires sufficient memory and storage for subsystems.

**New:** The RUV subsystem requires a minimum CSA storage of 568K for product modules.

**Changed:** The RUV ISPF interface requires the IBM Interactive System Productivity Facility (ISPF) version 4.2 or later.

**Obsolete:** None
Rule Sets

Purpose: RUV uses rule sets to determine which VSAM files should be protected.

New: The ADD USER_VARIABLES command creates user variables.

The ADD JOURNAL_MODEL command defines log and journal models named in the job set rule.

The ADD JOB_SET command identifies the user’s jobs that will trigger batch journaling activity.

The ADD RULE_SET command identifies the set of rules that combines all the options a user has defined to control the batch journaling environment.

The ADD VSAM_SET command identifies the recoverable VSAM files by DSN and the type of recovery to support at the DSN level.

The ADD JOB_JCL creates a recovery JCL record to be used in the event of a user abend.

The ADD INTERNAL_READER command creates a rule set element that controls the processing of recovery JCL that is generated when a job abends.

The ISPF online interface and the ACTIVATE RULE_SET command activates a rule set.

Changed: None

Obsolete: None

Security

Purpose: RUV uses APF interfaces to allow flexibility for the user to implement security.

New: RUV has specific security parameters in addition to standard site security.

Changed: The product load library must be APF-authorized.

Obsolete: None
ISPF Interface

Purpose: The ISPF interface allows you to perform a VSAM recovery, display backup and archive data sets, display a log of logs list, display subsystem and rule set information and create rule sets, and look up message information.

New: The ISPF interface can be used as an automated recovery tool, and provides online interaction for repository data.

Changed: None

Obsolete: None

Using BMC Software Subsystems

Purpose: BMC Software subsystems activate or facilitate RUV and other BMC Software products.

New: The BMC Software Primary Subsystem (BMCP) and the BMC Software Consolidated Subsystem (BCSS) are now required to implement and use the RUV and to access the repository.

Changed: In RPCV, the BMCP and BCSS subsystems were required only when the customer implemented the JOB STEP logging and backout features.

Use the SET SUBSYSTEM command to connect a batch job to a specific subsystem.

Obsolete: The RPCV subsystem and the Backup Subsystem (RPCS) are not valid for RUV product.

Batch Journaling and Job Step Logging

Purpose: Provides recovery data for batch processing.

New: Batch journaling requires a rule set to be defined in the RUV repository. The rule set contains all of the information that is necessary to indicate what VSAM files will be protected (journaled or logged). The default rule set is #DEFAULT. This member is located in the RUV.CNTL library.

The BMCP and the BCSS are now required for this feature.

Program RUVZCJC converts an ISIJCT to RUV rule set.
Changed: One output journal or log is created per job step. Dynamic backout is provided if the proper recovery JCL is defined in the rule set and logging is requested for the step.

Obsolete: Programs ISIREP and ISI0192A are no longer required.

The ISIOPT, ISIJCT, and ISICOPT tables are no longer required.

The RPCV subsystem option tables (global option, write-ahead log, backout exclude data sets, and initialization list) are no longer required. An equivalent program to perform a backout to the beginning of a job step has been dropped.

---

**Backing Up Master Data Sets**

**Purpose:** The backup command writes one or more copies of a VSAM base cluster to sequential data set(s) in RUV prefixed format.

**New:** The backup command has a new syntax. The registration of the backup is done automatically. The BMCP and the BCSS are now required for this feature and for the Backup-While-Open (BWO) feature. The CICS file definition for VSAM files that are to be processed through the BWO feature must include the update attribute.

**Changed:** Program RUVZSM0 is used to process the Backup command.

The CICS RDO entries for PLTPI and PLTSD have changed. See members RUVPLTPI and RUVPLTSD of the RUV.CNTL library. If you plan on using BWO, RUVB8020 is required in the PLTPI and PLTSD. This is the BWO CICS Controller, which communicates with the RUV subsystem.

The setting of the CICS transient data destination ID used for Subsystem Backup information messages has a default setting of “CSMT”.

The setting of the first three characters of the RUV CICS transaction IDs has changed. The default transaction ID prefix is “RUV.”

Use the SET SUBSYSTEM command to connect a batch job to a specific subsystem when performing backup (or BWO) so the registration is done in the correct repository.

**Obsolete:** The following feature has been dropped:

- CIMODE
Purge Processing

Purpose: The PURGE command purges obsolete repository records.

New: None

Changed: The number of days and cycles per file can be set using the UPDATE/ADD VSAM_FILE commands.

Obsolete: None

Archiving with RUV

Purpose: The ARCHIVE command will register recovery information in the repository.

New: The ARCHIVE command causes a CICS journal or log stream to be read and reformatted into RUV archive records.

Changed: Program RUVZSM0 is used to process the Archive command.

Obsolete: The JOURNAL command is obsolete.

RUV Recovery

Purpose: Recover VSAM files in parallel.

New: The RESTORE command has a complete new syntax. The RECOVER command is used to perform recover forward or backout.

The RUV recovery processing performs parallel restore, recovery, and BLDINDEX tasks. Use the Set READER_TASKS(m) command to control the maximum number of parallel tasks per command.

RUV product recovery processing restores the latest RUV or SMS backup master file.

The RESTORE VSAM_FILE command restores VSAM master file from the latest RUV or SMS backup master file in preparation for a forward recovery.

The RECOVER FORWARD command performs a forward recovery.

The RECOVER BACKOUT command performs a backward recovery.
Creating RUV Reports

**Purpose:** The REPORT command creates a report on data in the repository.

**New:** RPCV-created journals and logs can be processed by the RUV utilities, but they must be registered by the ARCHIVE or REGISTER command.

The reports have a new format. You can use the SET command with the REPORT option to set the reporting level (SUMMARY | DETAIL | FULL).

**Changed:** None

**Obsolete:** None

Miscellaneous Items

**Purpose:** Many features have been augmented with additional functionality or changes.

**New:** Most commands allow wildcards.

**Changed:** The group feature lets you process many files at the same time and in the same way. For example, you can recover related files as a group rather than individually.

**Obsolete:** In-flight processing is supported as a result of changes in the CICS Transaction Server. See “Recovery Process in RUV” on page 46
This appendix provides information-gathering work sheets. The work sheets can help you solve problems that may occur as you construct rule sets. If you use these work sheets, you will maintain a written record of the VSAM_SET, JOB_SET, and journal model definition data you have developed.

Make copies of each form. Date the form and write the specific name of the VSAM_SET, JOB_SET, and journal model definition. Fill in appropriate blocks.

For information about rule sets, see Chapter 4, “Using the Batch Journaling Facility.”

Use the VSAM_SET form to record VSAM DSNs and related information. You generally group VSAM rules together under one or more VSAM sets. To make VSAM files recoverable, you use VSAM rules. VSAM rules identify recoverable VSAM data sets and specify the type of recovery that can be performed against those data sets.

Use the JOB_SET form to record each job name and related information. You generally group job rules together under one or more job sets. To identify the jobs and programs that use recoverable VSAM files, and to identify the recovery actions you want RUV to take if an abend occurs, you use job rules. Job rules control when RUV logging and journaling occur by job and by program, as well as where RUV writes the log and journal information.

Use the Journal Model Definition form to record each journal model name and related information. You use journal models to create templates that will construct a DD allocation statement for QSAM logs and journals. RUV uses logs to record information needed to perform step backout and uses journals for forward recovery (long-term retention of data).
<table>
<thead>
<tr>
<th>VSAM DSN</th>
<th>Protect/Exclude</th>
<th>Backout Recovery</th>
<th>Forward Recovery</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job Name</td>
<td>Program Name</td>
<td>Forward Recovery</td>
<td>Backout Recovery</td>
<td>Journal Abend</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>------------------</td>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No/Yes Model:</td>
<td>No/Yes Model:</td>
<td>Abend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JCL:</td>
<td>JES/PDS:</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abend</td>
<td>Continue</td>
<td></td>
</tr>
<tr>
<td>Journal Model Name</td>
<td>Activity Estimate</td>
<td>DSN</td>
<td>DASD Space/Tape</td>
<td>Remarks</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
<td>-----</td>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Coordinated Recovery Process

This appendix describes a process that you can use to coordinate recovery between IMS and DB2 systems or IMS, DB2, and VSAM systems. The content of this appendix is based on the white paper “A Coordinated Recovery Process: Obtaining Outage Free Recovery Points for IMS and DB2 Disaster Recovery” that is published by BMC Software. The original white paper, which includes sample REXX code for automating portions of the coordinated recovery process, is available on the BMC Software Web site.

**WARNING**

The REXX code and the processes that are described in this appendix are delivered as is, and no warranties, expressed or implied, are provided with these shareware solutions.

The following topics are included in this appendix:

- The Requirement for Coordinated Recovery Management ........................................... 666
- COORDINATED RECOVERY MANAGER ................................................................. 666
- An Alternative—Outage-Free Coordinated Recovery ................................................. 667
- A Coordinated Recovery Process for IMS and DB2 ................................................. 668
- Overview .................................................................................................................. 668
- Details of the Coordinated Recovery Process for IMS and DB2 ............................... 669
- Adding CICS/VSAM to the Coordinated Recovery Process .................................... 674
- Assumptions About the IMS, DB2, and VSAM Processes ........................................ 674
- The DB2 and VSAM Processes Without IMS ............................................................ 675
- The IMS, DB2, and VSAM Processes ........................................................................ 675
- The REXX EXECs ...................................................................................................... 679
- The IMS and DB2 REXX EXEC—CRPREXX ............................................................ 679
- The VSAM, DB2, and IMS REXX EXEC ................................................................. 680
The Requirement for Coordinated Recovery Management

In many large data processing organizations, applications have evolved and established relationships between IMS databases, DB2 databases, and CICS/VSAM files. Sometimes these relationships are known to the application (as in the case of a two-phase commit process), but frequently it is a more casual relationship. A transaction in one database management system may spawn a subsequent transaction in a different database management system. The transactions are tied, but the log data is not coordinated.

This situation requires a coordinated recovery between IMS, DB2, and CICS/VSAM applications for both local recovery and disaster recovery. To ensure data integrity, transactions and batch jobs on all systems are drained or stopped to establish a system-wide point of consistency (SWPOC). This technique is preferable from a data integrity standpoint, but it causes an availability impact that is in conflict with an environment that must be available 24 hours a day, seven days a week.

COORDINATED RECOVERY MANAGER

BMC Software provided a COORDINATED RECOVERY MANAGER (CRM) facility between 1996 and 2001. This facility was delivered at no additional charge with the RECOVERY MANAGER (RMGR) products for IMS, DB2, and OS/390. CRM was shareware that allowed the user to associate groups from the RMGR products and conduct operations on the resulting larger CRM group. CRM was a useful way to gather together a large number of disparate database objects and consolidate them into a single manageable entity.

The supported operations that were performed on the CRM group included a Hold Point of Consistency (HPC) function, which established a SWPOC by causing and coordinating IMS /DBR and DB2 STOP commands along with VSAM file deallocations. The resulting recovery point was registered in the CRM repository and used by future recovery events to create a massive point-in-time (PIT) recovery with assured data integrity.

Unfortunately, obtaining a SWPOC required a lengthy outage that most environments could not tolerate. These environments required an alternative solution. Some have reviewed advanced-technology solutions such as disk mirroring or shadowing, log capture with forward recovery, and remote-site tape vaulting. All of these solutions carry a cost-and-risk burden that renders them ineffective in many environments.
BMC Software offers an alternative process for obtaining a recovery point. This process ensures data integrity after system-wide recovery, yet it requires no outage to establish a SWPOC. This unique process is built on the intelligence and innovation in the BMC Software recovery products for IMS, DB2, and CICS/VSAM. This process, as described in this document, replaces the use of CRM for coordinated recovery. BMC Software dropped support for CRM in June 2001.

An Alternative—Outage-Free Coordinated Recovery

Using exclusive technology, it is possible to capture information from the database repositories and use it to derive a consistent and coordinated recovery point. The probable use for this process is in support of disaster recovery (DR), but some applications (for example, large ERP applications with correlated legacy applications) may make use of the procedure for local recoveries.

The following sections explain some examples of exploiting technologies to create a coordinated recovery point. The first section describes the process of taking an IMS log switch timestamp and using it to create a recovery point at that timestamp for both IMS and DB2 applications. The second section provides the same type of example for a DB2, IMS, and VSAM shared environment.

NOTE

Use the processes described in this paper as examples. Your environment probably has standards or procedures that will require some local modifications to these processes.
A Coordinated Recovery Process for IMS and DB2

The following section describes a coordinated recovery process for use in IMS and DB2 shared environments.

Overview

The coordinated recovery process for IMS and DB2 consists of the following major tasks:

1. Use the BMC Software RECOVERY MANAGER (RMGR) for IMS Disaster Recovery RECON Cleanup (DRRCN) utility to report on the RECONs, looking for the latest, safest timestamp that can be used in a point-in-time (PIT) recovery. This timestamp is the STOPTIME of the oldest SLDS (the one with the least recent timestamp) in an IMS data-sharing environment. RMGR for IMS ensures that all open logs are properly handled before choosing the PIT timestamp.

2. Pass the captured timestamp to the RMGR for DB2 Timestamp Insertion (ARMBTSI) batch program, which inserts the timestamp into the RMGR CRRDRPT repository table.

3. Execute the RMGR for DB2 Coordinated Disaster Recovery (ARMBCRC) batch program, which formats the timestamp into a relative byte address (RBA) or, in a DB2 data-sharing environment, a log record sequence number (LRSN). The RBA or LRSN is used to generate a DB2 conditional restart control record (CRCR) to that RBA or LRSN.

4. Execute the RMGR for DB2 System Resource Recovery (ARMBSRR) batch program, which generates a DB2 subsystem recovery and restart based on the CRCR RBA or LRSN that was built by the ARMBCRC program in the prior step. The DB2 restart process backs out any in-flight transactions at the RBA or LRSN.

5. Pass the latest IMS timestamp to RMGR for IMS to generate one or more jobs that execute the BMC Software Recovery utility for IMS (which is available with the Backup and Recovery Solution for IMS product and the RECOVERY PLUS for IMS product) and that recover the databases to this timestamp. Transactions that were in-flight at the timestamp are not applied during the recovery.

NOTE

This process assumes that the IMS and DB2 systems share a sysplex timer; otherwise, the timestamps will not coincide.
The net effect to your application is just as if the power was dropped at the local site at the time represented by the PIT timestamp and you restarted all the IMS and DB2 subsystems. You would lose the work that was in-flight at the time of the power failure. You would be at the same position at the DR site. You can use the RMGR for IMS Log Analysis function to obtain a report of work that was in progress at the timestamp you used for recovery.

The process flow includes application copies before the coordinated recovery process and dumps of the appropriate system files (catalogs, directories, libraries, and so on) after the coordinated recovery process.

**Details of the Coordinated Recovery Process for IMS and DB2**

The following sections explain the coordinated recovery process for IMS and DB2 in detail.

**Preparation Steps**

Perform the following preparation steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prepare the DR site.</td>
<td>Ensure that valid copies of the IMS and DB2 system libraries and BMC Software RMGR product libraries are available at the DR site.</td>
</tr>
<tr>
<td>2</td>
<td>Create and validate RMGR groups.</td>
<td>Create and populate RMGR groups to be used for possible DR scenarios. Use the Group Validation function regularly to ensure that groups are kept updated with changes in database allocations.</td>
</tr>
<tr>
<td>3</td>
<td>Create an RMGR profile for DR.</td>
<td>Create an RMGR for IMS DR profile that is tailored for usage at the DR site (for example, turn on secondary image copies and secondary logs for use in recovery, check output devices that are used for change accumulation, review SORT parameters, and so on).</td>
</tr>
<tr>
<td>4</td>
<td>Check recovery assets.</td>
<td>Periodically (perhaps monthly) verify the DR process by performing the Check Assets function (to ensure availability of recovery assets) and by using the Create Recovery JCL function (to create practice JCL).</td>
</tr>
</tbody>
</table>
At the Local Site

On a periodic basis (perhaps nightly), perform the following preparation steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Copy objects.</td>
<td>Copy the IMS and DB2 application objects. (You can create clean or fuzzy copies; it does not matter.)</td>
</tr>
<tr>
<td>2</td>
<td>Capture allocation information.</td>
<td>Run the RMGR for IMS Automatic Delete/Define (DRAMS) utility Capture function to store allocation information about database data sets in the RMGR for IMS repository.</td>
</tr>
<tr>
<td>3</td>
<td>Back up system objects.</td>
<td>Back up the IMS RECONs, DB2 catalog and directories, all BMC Software RMGR product repositories, and all IMS and DB2 system libraries and data sets. Prepare these backups for shipment to the remote site.</td>
</tr>
<tr>
<td>4</td>
<td>Create system log data sets.</td>
<td>Switch the IMS online log data sets (OLDSs) to create system log data sets (SLDSs) on tape, and prepare them for shipment.</td>
</tr>
<tr>
<td>5</td>
<td>Generate the DR timestamp.</td>
<td>Execute the RMGR for IMS RECON Cleanup (DRRCN) utility Check function to generate a report that contains the LATEST PIT RECOVERY TIME IS: value field. The value is the coordinated recovery disaster recovery timestamp (CR DR TS).</td>
</tr>
<tr>
<td>6</td>
<td>Conduct date/time conversion.</td>
<td>Invoke the CRPREXX REXX EXEC program to browse the DRRCN report output and conduct date/time conversion on the CR DR TS. Capture the timestamp value and ship it to the remote site with the other recovery assets.</td>
</tr>
<tr>
<td>7</td>
<td>Pass the converted timestamp value to ARMBTSI</td>
<td>Pass the converted CR DR TS value to the RMGR for DB2 Timestamp Insertion (ARMBTSI) batch program, which inserts a row into the CRRDRPT repository table.</td>
</tr>
<tr>
<td>8</td>
<td>Translate the timestamp to an RBA/LRSN value.</td>
<td>Invoke the RMGR for DB2 Coordinated Disaster Recovery (ARMBCRC) batch program, which takes the timestamp value from the CRRDRPT table and translates it into the proper RBA/LRSN value. This value becomes the input to the Conditional Restart Control Record (CRCR) process that is used later by the RMGR for DB2 System Resource Recovery (ARMBSRR) batch program.</td>
</tr>
<tr>
<td>9</td>
<td>Archive the active DB2 logs.</td>
<td>Archive the active DB2 logs through the RMGR for DB2 Archive Log Creation (ARMBLOG) batch program.</td>
</tr>
</tbody>
</table>
Figure 163 shows the process flow at the local site.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Copy the archive logs.</td>
<td>Copy the archive logs. Invoke the RMGR for DB2 Archive Log Data Sets (ARMBARC) batch program to copy the logs that are needed for recovery, and ship the copies to the remote site. ARMBARC also serves the very useful purpose of capturing the image copy information for the three “special” DB2 Catalog/Directory objects (DSNDB01.DBD01, DSNDB01.SYSUTILX, and DSNDB06.SYSCOPY). These objects are fundamental to the success of DB2 Disaster Recovery, and image copy information is not stored in the normal fashion on DSNDB06.SYSCOPY.</td>
</tr>
<tr>
<td>11</td>
<td>Generate JCL to rebuild DB2 at the remote site.</td>
<td>Pass the CRCR RBA/LRSN value to the ARMBSRR batch program, which generates the 200+ job steps to rebuild DB2 at a remote site to the designated RBA/LRSN (the CRCR established by the ARMBCRC program). The ARMBSRR program also generates recoveries for the RMGR for DB2 repository tables. Ship the output of the ARMBSRR program to the remote site.</td>
</tr>
<tr>
<td>12</td>
<td>Generate JCL to recover application data.</td>
<td>Invoke the RMGR for DB2 Backup and Recovery JCL (ARMBGEN) batch program to generate recovery JCL for the application data. Ship the output to the remote site.</td>
</tr>
<tr>
<td>13</td>
<td>Back up the ICF catalog and TMS directory.</td>
<td>Back up the operating system catalog (ICF) and the Tape Management System (TMS) directory, and prepare the backups for shipment to the remote site.</td>
</tr>
</tbody>
</table>
At the Remote Site

When you are performing disaster recovery at the remote site, the process is relatively straightforward. After the operating system has been restored, you simply restore the IMS RECONs and prepare them for a cold-start of IMS, release the jobs that are generated by the ARMBSRR and ARMBGEN programs, and generate IMS database recoveries to recover the data to the CR DR TS that was captured by the DRRCN utility and that was used by the CRPREXX program. Any in-flight IMS or DB2 transactions that were active at the time of the CR DR TS are not recovered.

Perform the following steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Restore IMS system and product libraries.</td>
<td>Restore IMS RECON (RECON1) and system libraries, as well as the RMGR for IMS data sets.</td>
</tr>
<tr>
<td>2</td>
<td>Clean up the RECON.</td>
<td>Execute the RMGR for IMS RECON Cleanup (DRRCN) utility Update function to prepare the IMS RECON data sets for startup. Make a note of the value that the utility reports in the LATEST PIT RECOVERY TIME IS: value field.</td>
</tr>
<tr>
<td>3</td>
<td>Allocate and REPRO RECONs.</td>
<td>Allocate RECON2 and RECON3 data sets, and REPRO the cleaned RECON1 to RECON2.</td>
</tr>
<tr>
<td>4</td>
<td>Perform IMS startup.</td>
<td>Cold-start IMS, and start RMGR for IMS.</td>
</tr>
<tr>
<td>6</td>
<td>Generate PIT recovery JCL for IMS objects.</td>
<td>Use the value from the LATEST PIT RECOVERY TIME IS: value field (from the execution of the RECON Cleanup utility at the local site); this value is also the CR DR TS that is used in the CRPREXX program. Enter this value in the RMGR for IMS ISPF dialog to generate the appropriate BMC Software Recovery utility JCL to conduct a PIT recovery to that timestamp.</td>
</tr>
<tr>
<td>7</td>
<td>Release PIT recovery JCL.</td>
<td>Release the generated IMS PIT recovery jobs (the PIT timestamp is the time that was captured in the RECON Cleanup utility previously in this procedure).</td>
</tr>
<tr>
<td>8</td>
<td>Recover the DB2 system.</td>
<td>Execute the jobs that the ARMBSRR program generates to recover the DB2 system and RMGR for DB2 repository. (The two jobs are stored in one data set). Remember to specify the DEFER ALL parameter in the ZPARMS for the remote site startup; if you do not, disk allocation failures occur as DB2 tries to mount volumes to back out in-flight transactions on data that doesn’t exist yet.</td>
</tr>
<tr>
<td>9</td>
<td>Recover DB2 applications.</td>
<td>Release the DB2 application recovery jobs that are generated by the ARMBGEN program, the remote site, or both.</td>
</tr>
</tbody>
</table>
Figure 164 shows the processes at the remote site. After the operating system and libraries are restored, the IMS and DB2 activities can run in parallel as long as enough resources are available. CR DR PiT TS is the coordinated recovery point.

Figure 164  Coordinated Recovery Process at the Remote Site
Adding CICS/VSAM to the Coordinated Recovery Process

This section describes the processes required to implement a straightforward recovery approach to include VSAM files in a coordinated IMS, DB2, and VSAM disaster recovery plan. This approach results in the ability to generate a coordinated recovery state across the three databases for the covered applications with only committed units of work in the recovered databases and files. Consideration has been given in this approach to making the process as automated as possible and able to be scheduled on a daily basis.

Assumptions About the IMS, DB2, and VSAM Processes

The IMS, DB2, and VSAM processes make the following assumptions:

- You are running the Recovery Utility for VSAM (RUV) product, version 4.1 or later, at the local and DR sites.
- You are using the RUV Batch Journaling function and retaining the logs, or you are running RUV backups after batch updates.
- You are using standard RUV backups.
- You are using Transaction Server 3.1 or later.
- The coordination driver is IMS in this example. The process is a little different if you are using DB2 and VSAM without IMS.
- The following processes include sample JCL references, which are included at the end of this document. These samples include the RUV commands needed to implement the process, but these commands must be modified to fit your particular implementation standards and requirements and require additional RUV configuration (such as defining your VSAM files, groups, and output models).
The DB2 and VSAM Processes Without IMS

If you want to implement a DB2 and VSAM coordinated recovery approach without using IMS, you could model the approach on the IMS, DB2, and VSAM processes by changing the coordination driver to DB2. For example, the following general tasks would need to be performed:

- Perform the DB2 log switch, and verify that the log switch has completed.
- Get the DB2 log switch timestamp (RBA).
- Convert the DB2 timestamp from local time to GMT.
- Convert this GMT timestamp to the RUV hexadecimal timestamp.
- Recover the files with RUV.

The IMS, DB2, and VSAM Processes

The following sections describe the processes you would perform to add VSAM to the overall IMS and DB2 disaster recovery plan.

At the Local Site

Perform the following steps at the local site. The first five steps are VSAM-only processes. The sixth and seventh steps are modifications to the existing processes that are required for IMS-and-DB2-only recovery.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Perform the RUV backup process.</td>
<td>The RUV backup process creates local and offsite backup stacked tapes and registers them in the RUV repository. Files can be specified individually or by group (using the RUV VSAM Group feature). This process is usually scheduled to run based on processing windows; it might run weekly or daily. Note: In this example, files that are destined to be shipped offsite are created initially with a location named “Staging” and will be updated to a location named “Offsite” after pull list reports have been run just prior to the shipment.</td>
</tr>
</tbody>
</table>
### The IMS, DB2, and VSAM Processes

#### Recovery Utility for VSAM User Guide

1. **Perform the RUV archive process.**
   - The RUV archive process extracts recovery data from the system and user logs, formats the data into the RUV archive format, and registers the archives in the RUV repository.
   - This process is usually scheduled to run based on transaction volume; it might run hourly or several times per day.

2. **Perform the RUV archive accumulation process.**
   - The RUV archive accumulation consolidated process combines the archives from multiple archive processes into a single consolidated archive. This process is performed for the user data and for the unit of work data. If the RUV Batch Journaling feature is being used and logs are available, these logs are included.
   - This process creates local and offsite copies and should be scheduled to complete shortly before the daily shipment to the offsite location.
   - **Note:** If you are running batch updates to your VSAM files, you are not using batch journaling, and you are not retaining the logs, you must run RUV backups to ensure that all updates are captured and available offsite.

3. **Perform the RUV repository backup process.**
   - The RUV repository backup process creates a local copy and an offsite copy of the RUV repository. The repository contains registration information for your files, groups, backups and archives. The offsite copy should be sent to the DR site along with the backups and archives.
   - This process should be scheduled to complete shortly before the daily shipment to the offsite location.

4. **Perform the RUV pull list process.**
   - The RUV pull list process runs a Backup report for all backups and archives in the “Staging” location, and then it changes the location for those backups and archives to the “Offsite” location. The files that are listed in these reports should be shipped to the offsite location.
   - This process should be scheduled to complete shortly before the daily shipment to the offsite location.

5. **Generate the PIT using the REXX EXEC.**
   - Run the RECOV.REXX script to extract the recovery point from the IMS RECON report, format the recovery point into the appropriate DB2 and VSAM recovery formats, and update DB2 and VSAM recovery JCL.
   - This process should be scheduled to complete shortly before the daily shipment to the offsite location.

6. **Backup Recovery JCL Library**
   - Take a backup of the recovery JCL PDS file and ship to the offsite location along with the IMS, DB2 and VSAM backups and archives/logs.
Figure 165 shows the IMS, DB2, and VSAM components of the coordinated recovery process at the local site.

**Figure 165  Coordinated Recovery at the Local Site (IMS, DB2, and VSAM)**

### At the Remote Site

When a disaster is declared (or simulated), perform the following steps to recover the VSAM files:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Restore recovery JCL.</td>
<td>Restore the recovery JCL that was created in Step 6 of the local site process.</td>
</tr>
</tbody>
</table>
| 2    | Catalog data sets.              | Catalog the backups, archives, and repository backup that were created in Steps 2, 3, and 4 of the local site process.  
Technical, it is not required for these data sets to be cataloged. If they are not cataloged, RUV calls for them by VOLSER during recovery. |
| 3    | Start the RUV subsystem.        | Start the RUV subsystem.                                                                                                                                 |
| 4    | Restore the RUV repository.     | Restore the RUV repository by using the RUVREPRS program.                                                                                   |
| 5    | Inactivate local files and activate offsite files. | Run the RUVTOGL program to inactivate local backups and archives and activate the offsite files.                                                  |
Figure 166 shows the IMS, DB2, and VSAM components of the coordinated recovery process at the remote site.

Figure 166  Coordinated Recovery at the Remote Site (IMS, DB2, and VSAM Involved)
The REXX EXECs

The REXX programs and process descriptions are delivered as is, and no warranties, expressed or implied, are provided with these shareware solutions. Use them at your own risk, and feel free to modify them as you see fit. Refer to the BMC Software product documentation sets for the most complete details about product usage and other tips.

The IMS and DB2 REXX EXEC—CRPREXX

A REXX program that captures the IMS recovery point timestamp and passes it to the ARMBTSI program is distributed in member IRMCRDB2 of the RMGR for IMS sample library. The high-level process flow is as follows:

1. Browse the output of the RMGR for IMS DRRCN utility.

2. Obtain the timestamp value from the LATEST PIT RECOVERY TIME IS: value field.

3. Convert the timestamp.

4. Present the converted timestamp to the RMGR for DB2 ARMBTSI program, which inserts the value into a repository table (CRRDRPT).

As written, the DRRCN report is expected to be in 'BMCIRM.BB.DRRCN.REPORT' and the local offset from GMT is expected to be in 'BMCIRM.BB.DRRCN.OFFSET'. The offset should be in the first three columns: +05, -11, and so on. The JCL for ARMBTSI is in 'BMCIRM.BB.DRRCN.ARMBTSI.CNTL' and the REXX routine updates the TIMESTAMP in the PARM string in place.

NOTE

The offset file 'BMCIRM.BB.DRRCN.OFFSET' is used only if option TIMEVALUE=GMT is specified (RMGR for IMS DRRCN utility). The default is to show all times in local format in which case the timestamp does not have to be adjusted.

You can modify the REXX as appropriate to your shop standards.

NOTE

The REXX EXEC CRPREXX examines the output of the RMGR for IMS DRRCN utility and is therefore dependent on the format and content of that output. Any implementation of this REXX EXEC should be re-examined, modified accordingly, and tested when any new release of the DRRCN utility is installed (the DRRCN utility is provided with RECOVERY MANAGER for IMS and the Backup and Recovery Solution for IMS product).
The VSAM, DB2, and IMS REXX EXEC

A REXX program that captures the IMS recovery point timestamp, passes it to the ARMBTSI program, and renders it valid for Recovery Utility for VSAM (RUV) is contained in member IRMCRRUV of the RMGR for IMS sample library. The high-level process flow is as follows:

1. Browse the output of the RMGR for IMS DRRCN utility.

2. Obtain the timestamp value from the LATEST PIT RECOVERY TIME IS: value field.

3. Convert the timestamp.

4. Present the timestamp to the RUV recovery job.

5. Present the converted timestamp to the RMGR for DB2 ARMBTSI program, which inserts the value into a repository table (CRRDRPT).

As written, the DRRCN report is expected to be in 'BMCIRM.BB.DRRCN.REPORT' and the local offset from GMT is expected to be in 'BMCIRM.BB.DRRCN.OFFSET'. The offset should be in the first three columns: +05, -11, and so on. The JCL for ARMBTSI is in 'BMCIRM.BB.DRRCN.ARMBTSI.CNTL' and the REXX routine updates the TIMESTAMP in the PARM string in place.

**NOTE**
The offset file 'BMCIRM.BB.DRRCN.OFFSET' is used only if option TIMEVALUE=GMT is specified (RMGR for IMS DRRCN utility). The default is to show all times in local format in which case the timestamp does not have to be adjusted.

You can modify the REXX as appropriate to your shop standards.

**NOTE**
The REXX EXEC CRPREXX examines the output of the RMGR for IMS DRRCN utility and is therefore dependent on the format and content of that output. Any implementation of this REXX EXEC should be re-examined, modified accordingly, and tested when any new release of the DRRCN utility is installed (the DRRCN utility is provided with RECOVERY MANAGER for IMS and the Backup and Recovery Solution for IMS product).
The following sections provide summary information about the Recovery Utility for VSAM product. The following topics are included:

Conventions .................................................. 682  
   Command Syntax Conventions ................................ 682  
   Masking (Wildcard) Characters .................................. 683  
   Keyword Variable Value Conventions ............................... 683  
RUVZSM0 Reference ........................................... 685  
   JCL to Execute RUVZSM0 ...................................... 685  
   Command Language Elements and Syntax ...................... 685  
   RUVZSM0 Commands .......................................... 686  
User Variables .................................................. 712  
   User-Defined Variables ....................................... 712  
   Substring Notation .......................................... 712  
   RUV-Defined Variables ....................................... 713  
   IBM Variable Substitution Symbols ............................. 715  
   RUV Action Variables ....................................... 715  
RUV Subsystem Commands ........................................ 717  
   BMCP Console Commands .................................... 717  
   BCSS Console Commands .................................... 717  
   BWO CICS Commands ........................................ 718  
User Exit Parameter List ........................................ 719
Conventions

The following sections explain the conventions that apply to the information in this appendix.

Command Syntax Conventions

This document uses the following conventions to show command syntax.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQUIRED</td>
<td>Required elements are shown in uppercase.</td>
</tr>
<tr>
<td>{ OPTIONAL }</td>
<td>Optional elements are enclosed in braces. Required elements are not enclosed in braces. Do not code braces in your own control statements.</td>
</tr>
<tr>
<td>[ REQUIRED OPTIONS ]</td>
<td>Brackets indicate a group of options; one of the options is required.</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>Default values are underscored; also, default values usually appear first in a series of values.</td>
</tr>
<tr>
<td>variable</td>
<td>Variable values are shown in lowercase italic type. Substitute a valid value as appropriate. For more information, see “Keyword Variable Value Conventions” on page 683.</td>
</tr>
<tr>
<td>( ELEMENT1</td>
<td>ELEMENT2 )</td>
</tr>
<tr>
<td>( ELEMENT1 ( value )</td>
<td>ELEMENT2 ( value ) }</td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate that you can repeat the preceding element or group of elements one or more times.</td>
</tr>
<tr>
<td>KEYWORD ( ... )</td>
<td>A keyword with ellipses in parentheses indicates that you can code multiple instances of the keyword (along with its values, accompanying keywords, and their values) on the command.</td>
</tr>
<tr>
<td>'embedded blanks'</td>
<td>Apostrophes indicate a character string. Code the apostrophes as shown.</td>
</tr>
<tr>
<td>&quot;or 'embedded’ quotes&quot;</td>
<td>&quot;or 'embedded’ quotes’</td>
</tr>
<tr>
<td>KEYWORD ( value )</td>
<td>Parentheses enclose a keyword value. Code parentheses as shown.</td>
</tr>
<tr>
<td>* as in KEYWORD ( value, * , ... )</td>
<td>An asterisk indicates that you can code masking characters for this keyword on the indicated command. Valid masking characters are an asterisk ( * ) and a question mark ( ? ). For more information, see “Masking (Wildcard) Characters” on page 683.</td>
</tr>
</tbody>
</table>
Masking (Wildcard) Characters

In some keyword values for RUVZSM0, you can use the following masking (wildcard) characters.

<table>
<thead>
<tr>
<th>Character</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>? (question mark)</td>
<td>Matches any single character in the position in which you coded the question mark. For example, code AB?12? to select ABC123 and ABD124 but exclude AAB123 and AB1111.</td>
</tr>
<tr>
<td>* (asterisk)</td>
<td>Matches one or more characters starting at the position in which you coded the asterisk. For example, code ABC* to select ABC, ABC123, and ABC12345 but exclude AABC and AB.</td>
</tr>
</tbody>
</table>

Keyword Variable Value Conventions

This document uses the following conventions to indicate keyword variable values. The lengths noted are the maximum accepted lengths; the length of the value you code can be less.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>applid8</td>
<td>8-character name associated with a VTAM application (for example, a CICS region)</td>
</tr>
<tr>
<td>comment_fields</td>
<td>four 50-character fields available for user free-form comments</td>
</tr>
<tr>
<td>ddn8</td>
<td>8-character ddname of a DD statement</td>
</tr>
<tr>
<td>devicename8</td>
<td>z/OS esoteric name used for device allocations</td>
</tr>
<tr>
<td>dsn.variables</td>
<td>data set name used in job JCL that may contain RUV system or user variables</td>
</tr>
<tr>
<td>dsn44</td>
<td>fully qualified 44-character data set name</td>
</tr>
<tr>
<td>dsn_mask44</td>
<td>44-character data set name that can contain wildcard character(s)</td>
</tr>
<tr>
<td>fileid8</td>
<td>8-character file name</td>
</tr>
<tr>
<td>job_mask8</td>
<td>job name as referred to by z/OS JCL; can contain wildcard character(s)</td>
</tr>
<tr>
<td>job_name8</td>
<td>8-character job name</td>
</tr>
<tr>
<td>journal_model_name32</td>
<td>32-character data set name for model to be used when allocating a log file</td>
</tr>
<tr>
<td>literal</td>
<td>value inserted for test purposes only in the PARM keyword</td>
</tr>
<tr>
<td>location_name8</td>
<td>8-character field of user information that may denote location of the media</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lsn26</td>
<td>fully qualified 26-character z/OS log stream name</td>
</tr>
<tr>
<td>n1, n2</td>
<td>condition codes used in SET TERM_CC command</td>
</tr>
<tr>
<td>name8</td>
<td>8-digit member name used by the Batch Journaling Facility</td>
</tr>
<tr>
<td>name32</td>
<td>32-character user-defined name</td>
</tr>
<tr>
<td>nn, nnn, nnnn, . . .</td>
<td>numeric keyword substitution values used with RUV keywords</td>
</tr>
<tr>
<td>pds.dsn.with.variables</td>
<td>name of a PDS data set that will receive JCL generated by RUV processing</td>
</tr>
<tr>
<td>ppp</td>
<td>3-digit primary space allocation</td>
</tr>
<tr>
<td>program8</td>
<td>8-character name of a user program</td>
</tr>
<tr>
<td>program_mask8</td>
<td>8-character program name that can contain wildcard character(s)</td>
</tr>
<tr>
<td>ssid</td>
<td>4-character subsystem name set during RUV installation</td>
</tr>
<tr>
<td>sss</td>
<td>3-digit secondary space allocation</td>
</tr>
<tr>
<td>ttt</td>
<td>3-character type of space being requested; can be CYL, TRK, or BLK</td>
</tr>
<tr>
<td>var_permitted</td>
<td>variable for an RUV command that may be an RUV system or user variable</td>
</tr>
<tr>
<td>variable_name</td>
<td>part of the definition of an RUV user variable that defines the variable name</td>
</tr>
<tr>
<td>variable_value</td>
<td>part of the definition of an RUV user variable that sets the variable value</td>
</tr>
<tr>
<td>&lt;vvvv&gt;</td>
<td>value that may be specified by using an RUV system or user variable</td>
</tr>
<tr>
<td>* or wildcards</td>
<td>wildcard symbols ? or * to facilitate a search</td>
</tr>
<tr>
<td></td>
<td>? will match any one positional character.</td>
</tr>
<tr>
<td></td>
<td>* will match all characters from the asterisk position to the right.</td>
</tr>
<tr>
<td>xxx</td>
<td>3-digit number to indicate National Language support</td>
</tr>
<tr>
<td></td>
<td>Default is English.</td>
</tr>
<tr>
<td>xxxxx, . . .</td>
<td>character keyword substitution values used with RUV keywords</td>
</tr>
<tr>
<td>Xhhhhhhhhhhhhhhhh</td>
<td>timestamp expressed as 16 hexadecimal digits preceded by the character X</td>
</tr>
<tr>
<td></td>
<td>RUV expects all timestamps to be supplied in Greenwich Mean Time (GMT).</td>
</tr>
<tr>
<td>yyyyjjjhhmmst</td>
<td>yyyy is the 4-digit year</td>
</tr>
<tr>
<td></td>
<td>jjj is the 3-digit day of the year (1 to 366)</td>
</tr>
<tr>
<td></td>
<td>hh is the 2-digit hour of the day (00 to 23)</td>
</tr>
<tr>
<td></td>
<td>mmm is the 2-digit minutes of the hour (00 to 59)</td>
</tr>
<tr>
<td></td>
<td>ss is the 2-digit seconds (00 to 59)</td>
</tr>
<tr>
<td></td>
<td>t is the 1-digit tenths of the second (0 to 9)</td>
</tr>
<tr>
<td></td>
<td>RUV expects all timestamps to be supplied in (GMT).</td>
</tr>
</tbody>
</table>
RUVTZSM0 Reference

The following sections provide information about the Recovery Utility for VSAM (RUV) batch utility (program RUVTZSM0).

JCL to Execute RUVTZSM0

You can modify the following model JCL to execute RUVTZSM0:

```
// . . . job statement . . .
//RUVTZSM0 EXEC PGM=RUVTZSM0
//*STEPLIB DD ==>IF REQUIRED
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
RUVT control statements
//
```

Command Language Elements and Syntax

In RUVTZSM0 JCL, the SYSIN data set contains RUVT control statements, which are composed of the following command language elements. If RUVTZSM0 detects a syntax error, it issues a series of messages (to the SYSPRINT data set) that highlight the problem and document the correct syntax of the statement.

You can code any command element in uppercase, lowercase, or mixed case. Code command elements in columns 1–71.

<table>
<thead>
<tr>
<th>Element</th>
<th>Syntax Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>Code a command at the beginning of each control statement. The command is required and must precede all related keywords and values or variable operands.</td>
</tr>
<tr>
<td>keyword</td>
<td>Code required and optional keywords (in any order) after the command. You can omit keywords that have default values; RUVTZSM0 uses the default value automatically.</td>
</tr>
<tr>
<td>keyword values</td>
<td>Code one or more keyword values after a keyword as required (some keywords do not have values). Enclose the value in parentheses.</td>
</tr>
<tr>
<td>separators</td>
<td>Code a separator between multiple keyword/value combinations and between multiple keyword values. Valid separators are one or more blanks, a comma, a comma plus one or more blanks, a line break (new line), or a comma plus a line break.</td>
</tr>
<tr>
<td>command terminator</td>
<td>Code a semicolon (;) to designate the end of a command.</td>
</tr>
</tbody>
</table>
### RUVZSM0 Commands

The following commands are valid in RUVZSM0.

<table>
<thead>
<tr>
<th>Element</th>
<th>Syntax Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment</td>
<td>Code an asterisk (*) in column 1 to treat the entire line as a comment.</td>
</tr>
<tr>
<td>quote marks</td>
<td>Code quote marks (’ or ”) interchangeably to enclose a variable with embedded blanks or quote marks.</td>
</tr>
</tbody>
</table>

#### ACTIVATE RULE_SET (name32)
* activate a rule set

```bash
ACTIVATE RULE_SET ( name32 )

{ TRACE_ON }
```

#### ADD EXTERNAL_VENDOR (name32)
* create an external vendor definition

```bash
ADD EXTERNAL_VENDOR ( name32 )

{ COMMENT ( comment_fields ) }
{ BACKUP_REGISTRATION ( NO |

  program8 [, “initialization parameters” ]

)

{ JOURNAL_REGISTRATION ( NO |

  program8 [, “initialization parameters” ]

)

{ STATUS ( ACTIVE | INACTIVE ) }
```

#### ADD INTERNAL_READER (name32)
* create an internal reader definition to store JCL
* in the indicated PDS member

```bash
ADD INTERNAL_READER ( name32 )

{ COMMENT ( comment_fields ) }
PDS_OUT ( pds.dsn44.with.variables )
MEMBER ( var_permitted )
```

#### ADD INTERNAL_READER (name32)
* create an internal reader definition to write JCL
* to the indicated JES job class

```bash
ADD INTERNAL_READER ( name32 )

{ COMMENT ( comment_fields ) }
CLASS ( a )
```

#### ADD INTERNAL_READER (name32)
* create an internal reader definition to store JCL
* in the data set that has the indicated ddname

```bash
ADD INTERNAL_READER ( name32 )

{ COMMENT ( comment_fields ) }
DDNAME ( ddn8 )
```
### ADD JOB_JCL

* create a job JCL definition

```plaintext
{ COMMENT ( comment_fields ) }

; //<JOBNAME(1,8)> JOB (<ACCT>),CLASS=x, ...
//RUVZSM0 EXEC PGM=RUVZSM0
<RUV_STEPLIB> => IF REQUIRED
//SYSPRINT DD SYSOUT=* //SYSOUT DD SYSOUT=* //SYSIN DD *
<RUV_CONTROL>
<RUV_JOURNAL_DEACTIVATE>
END_DATA;
```

### ADD JOB_SET

* create a job set definition containing one or more job rule definitions

```plaintext
{ COMMENT ( comment_fields ) }

 JOB_RULE ( job_mask8, program_mask8, 
   { LOG ( EXCLUDE | name32 ) } 
   { JOURNAL ( EXCLUDE | name32 ) } 
   { JOB_ABEND ( SUBMIT | NOSUBMIT ) } 
   { JOB_JCL ( name32 ) } 
   { INTERNAL_READER ( name32 ) } 
   { JOURNAL_ABEND ( ABEND | CONTINUE ) } 
 )

{ JOB_RULE ( ... ) }

;```

### ADD JOURNAL_MODEL

* create a journal model definition as QSAM statements in JCL
* RUV variables are permitted

```plaintext
{ COMMENT ( comment_fields ) }

 PROTECTION_LEVEL ( 1 | 0 ) 
 QSAM (" DSN=xxxx.xxxx.xxxx," 
   " SPACE=(xxx,(xxx,xxx))," 
   " UNIT=xxxxx ), DISP=(xx) ")

;```

### ADD JOURNAL_MODEL

* create a journal model definition as the data set
* with the indicated ddname

```plaintext
{ COMMENT ( comment_fields ) }

 PROTECTION_LEVEL ( 1 | 0 ) 
 DDNAME ( ddn8 )

;```

### ADD LOG_OF_LOGS

* create repository information about the indicated log of logs file

```plaintext
{ COMMENT ( comment_fields ) }

 STATUS ( ACTIVE | INACTIVE )

;```
### RUVZSM0 Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD LOG_STREAM (lsn26)</strong></td>
<td><em>create repository information about the indicated log stream file</em></td>
<td>{ [ COMMENT(comment_fields) ] [ STATUS( ACTIVE</td>
</tr>
</tbody>
</table>
| **ADD OUTPUT_MODEL (name32)** | *create an output model definition* | { \[ COMMENT(comment_fields) \] \[ BACKUP_OUT_TYPE(DASD | TAPE | STACKED_TAPE) \] \[ STACKED_TAPE_COUNT(1 | nn) \] } *
| * QSAM statement as in JCL.* | |
| * RUV variables are permitted* | |
| * QSAM (" DSN=xxxx.xxxx.xxxx," |
| " SPACE=(xxx,xxx)," |
| " UNIT=xxxx, DISP=(xx) " ) } ; |
| **ADD RULE_SET (name32)** | *create a rule set definition* | { \[ COMMENT(comment_fields) \] \[ JOB_SET(name32, ...) \] \[ VSAM_SET(name32, ...) \] \[ EXTERNAL_VENDOR(name32, ...) \] \[ USER_VARIABLES(name32, ...) \] } ; |
| **ADD SEQ_FILE (dsn44)** | *create repository information about the indicated sequential file* | { \[ COMMENT(comment_fields) \] \[ SELECTION_EXIT(program8) \] \[ DAYS(0 | 45 | nnn | 999) \] \[ CYCLES(nnn) \] } ; |
| **ADD SEQ_GROUP_DEFINITION (name32)** | *create a sequential group definition* | { \[ COMMENT(comment_fields) \] \[ SEQ_FILE(dsn_mask44, |
| [ INCLUDE | EXCLUDE ] |
| ) \[ SEQ_FILE(...) \] } ; |
| **ADD USER_VARIABLES (name32)** | *create user variable definitions* | { \[ COMMENT(comment_fields) \] <variablename>/variablevalue/ END_DATA; |
ADD VSAM_FILE (dsn44)
* create repository information about the indicated VSAM file
  { COMMENT (comment_fields) }
  { SELECTION_EXIT (program8) }
  { APPLID_FILEID (applid8, fileid8) }
  { APPLID_FILEID (… ) }
  { DAYS (0 | 45 | nnn | 999 ) | CYCLES (nnn) }

ADD VSAM_GROUP_DEFINITION (name32)
* create a VSAM group definition
  { COMMENT (comment_fields) }
  VSAM_FILE (dsn_mask44,
    { INCLUDE | EXCLUDE }
  )
  { VSAM_FILE (… ) }

ADD VSAM_NAME_MODEL (name32)
* create a VSAM name model definition
  { COMMENT (comment_fields) }
  VSAM_FILE_MASK (dsn_mask44, dsn.variables)
    { REPLACE (NO | YES) }
    { HOSTCOPYMODE (SHARED | EXCLUSIVE) }

ADD VSAM_SET (name32)
* create repository information about the indicated VSAM set
* EXCLUDE is not valid with VSAM_GROUP on this command
  { COMMENT (comment_fields) }
  VSAM_GROUP (name32
    { BOTH | LOG | JOURNAL }
  )
  { VSAM_GROUP (… ) }
  VSAM_RULE (dsn44,
    { EXCLUDE | BOTH | LOG | JOURNAL }
  )
  { VSAM_RULE (… ) }

ARCHIVE
* create one or more accumulation archives
* at least one ACCUM keyword is required
  { INVENTORY_ONLY ( NO | YES )
  { ARCHIVE_IN ( ddn8_or_dsn44, * , ... )
  { SCRATCH ( NO | YES | FORCE )

ACCUM ( CONSOLIDATED
* you can code VSAM_FILE, VSAM_GROUP,
* VSAM_GROUP and VSAM_FILE,
* or UNSELECTED_VSAM_FILES
  { { VSAM_FILE ( ddnn8_or_dsn44, * , ... )
    { VSAM_GROUP ( name32, *, ... )
    | UNSELECTED_VSAM_FILES
  }
  { RETAIN_IMAGE ( BOTH | LOG | JOURNAL )
  { ARCHIVE_OUT ( ddnn8
    { COMMENT ( comment_fields )
    { LOCATION ( location_name8 )
    { STATUS ( ACTIVE | INACTIVE )
    { MUST_COMPLETE ( YES | NO )
  }}

ARCHIVE ( LAST_IMAGE
* you can code VSAM_FILE, VSAM_GROUP,
* or VSAM_GROUP and VSAM_FILE
  VSAM_FILE ( ddn8_or_dsn44, *, .. )
  VSAM_GROUP ( name32, *, .. )
  { RETAIN_IMAGE ( JOURNAL )
  } ARCHIVE_OUT ( ddn8
    { COMMENT ( comment_fields )
    { LOCATION ( location_name8 )
    { STATUS ( ACTIVE | INACTIVE )
    { MUST_COMPLETE ( YES | NO )
  }}

ARCHIVE ( ... )
ARCHIVE_ARCHIVE_IN (ddn8_or_dsn44)
* copy an archive file
   \{ SCRATCH ( NO | YES ) \}
ARCHIVE_OUT (ddn8
   \{ COMMENT ( comment_fields ) \}
   \{ LOCATION ( location_name8 ) \}
   \{ STATUS ( ACTIVE | INACTIVE ) \}
   \{ MUST_COMPLETE ( YES | NO ) \}

\{ ARCHIVE_OUT( ... ) \}

ARCHIVE_JOURNAL_IN (ddn8_or_dsn44)
* perform an archive of a journal
   APPLID (applid8)
   \{ COMPLETE_COPY \}
ARCHIVE_OUT (ddn8
   \{ LOCATION ( location_name8 ) \}
   \{ COMMENT ( comment_fields ) \}
   \{ STATUS ( ACTIVE | INACTIVE ) \}
   \{ MUST_COMPLETE ( YES | NO ) \}

\{ ARCHIVE_OUT( ... ) \}
\{ UNIT_OF_WORK_OUT (ddn8
   \{ LOCATION ( location_name8 ) \}
   \{ COMMENT ( comment_fields ) \}
   \{ STATUS ( ACTIVE | INACTIVE ) \}
   \{ MUST_COMPLETE ( YES | NO ) \}
\}

ARCHIVE_LOG_STREAM_IN (lsn26)
* perform an archive of a log stream
   \{ DELETE ( NO | YES ) \}
   \{ GAP ( STOP | WARN | IGNORE ) \}
   \{ COMPLETE_COPY \}
   \{ VIEW ( ACTIVE | ALL ) \}
ARCHIVE_OUT (ddn8
   \{ COMMENT ( comment_fields ) \}
   \{ LOCATION ( location_name8 ) \}
   \{ STATUS ( ACTIVE | INACTIVE ) \}
   \{ MUST_COMPLETE ( YES | NO ) \}

\{ ARCHIVE_OUT( ... ) \}
\{ UNIT_OF_WORK_OUT (ddn8
   \{ COMMENT ( comment_fields ) \}
   \{ LOCATION ( location_name8 ) \}
   \{ STATUS ( ACTIVE | INACTIVE ) \}
   \{ MUST_COMPLETE ( YES | NO ) \}
\}

;
ARCHIVE RPCV_IN (ddn8_or_dsn44)
* perform an archive of an RPCV journal
  { COMPLETE_COPY }
ARCHIVE_OUT (ddn8)
  { LOCATION (location_name8) }
  { COMMENT (comment_fields) }
  { STATUS (ACTIVE | INACTIVE) }
  { MUST_COMPLETE (YES | NO) }
}
ARCHIVE_OUT (...)
;

ARCHIVE SMF_IN (SYS1.MANx)
* perform an archive of repository records
ARCHIVE_OUT (ddn8)
  { LOCATION (location_name8) }
  { COMMENT (comment_fields) }
  { STATUS (ACTIVE | INACTIVE) }
  { MUST_COMPLETE (YES | NO) }
}
ARCHIVE_OUT (...)
;

ARCHIVE UNIT_OF_WORK_IN (ddn8_or_dsn44)
* copy a unit-of-work file
  { SCRATCH (NO | YES) }
UNIT_OF_WORK_OUT (ddn8)
  { STATUS (ACTIVE | INACTIVE) }
;

BACKUP REPOSITORY
* make a backup of the repository
BACKUP_OUT (ddn8)
  { COMMENT (comment_fields) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }
}
BACKUP_OUT (...)
{ SIMULATE }
;

BACKUP SEQ_GROUP (name32)
* make a backup of a group with sequential data sets
  { INVENTORY_ONLY (NO | YES) }
OUTPUT_MODEL (name32)
  { LOCATION (location_name8) }
  { COMMENT (comment_fields) }
  { STATUS (ACTIVE | INACTIVE) }
}
{ OUTPUT_MODEL (...)}
;
### BACKUP SEQ_IN (ddn8_or_dsn44)
* make a backup of a sequential data set
  { COMMENT ( comment_fields ) }
  { INVENTORY_ONLY ( NO | YES ) }
BACKUP_OUT (ddn8)
  { COMMENT ( comment_fields ) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE ) }

### BACKUP VSAM_GROUP (name32)
* make a backup of a group with VSAM data sets
  { INVENTORY_ONLY ( NO | YES ) }
  { PREFIX (YES | NO ) }
OUTPUT_MODEL (name32)
  { LOCATION (location_name8) }
  { COMMENT (comment_fields) }
  { STATUS (ACTIVE | INACTIVE ) }

### BACKUP VSAM_GROUP (name32)
* make a backup of a group with Instant Snapshot copies
  { INVENTORY_ONLY ( NO | YES ) }
VSAM_NAME_MODEL (name32)
  { LOCATION (location_name8) }
  { COMMENT (comment_fields) }
  { STATUS (ACTIVE | INACTIVE ) }
BACKUP VSAM_IN (ddn8_or_dsn44)
* make a backup of a VSAM data set
  { COMMENT (comment_fields) }
  { INVENTORY_ONLY(NO | YES) }
  { PREFIX(YES | NO) }
BACKUP_OUT (ddn8)
  { COMMENT (comment_fields) }
  { LOCATION(location_name8) }
  { STATUS (ACTIVE | INACTIVE) }
)
  { BACKUP_OUT( ... ) }
;
BACKUP VSAM_IN (ddn8_or_dsn44)
* make a backup of an Instant Snapshot copy
  { COMMENT (comment_fields) }
  { INVENTORY_ONLY(NO | YES) }
VSAM_NAME_MODEL (name32)
  { LOCATION(location_name8) }
  { COMMENT (comment_fields) }
  { STATUS (ACTIVE | INACTIVE) }
)
;
COPY VSAM_GROUP (name32)
* copy a group of Instant Snapshot copies with no registration
  { INVENTORY_ONLY(NO | YES) }
VSAM_NAME_MODEL (name32)
;
COPY VSAM_IN (ddn8_or_dsn44)
* copy an Instant Snapshot copy with no registration
  { COMMENT (comment_fields) }
  { INVENTORY_ONLY(NO | YES) }
VSAM_NAME_MODEL (name32)
;
DELETE APPLID (applid8,*,...)
* delete all repository information for a CICS APPLID
;
DELETE ARCHIVE_FILE (ddn8_or_dsn44,*,...)
* delete repository information for one or more archive files
  { FILTER_BY (}
    { LOCATION (location_name8) }
    { STATUS (ACTIVE | INACTIVE) }
  * if you omit FILTER_BY, default is to delete both active
  * and inactive statuses
    { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhh )}
    { STOP_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhh )}
  )
  { SCRATCH (NO | YES) }
;
DELETE BACKUP_FILE (ddn8_or_dsn44, *, ...)
* delete repository information for one or more backup files
  \FILTER_BY (  
    \LOCATION ( location_name8 )  
    \STATUS ( ACTIVE | INACTIVE )  
  )
* if you omit \FILTER_BY, default is to delete both active
* and inactive statuses
  \START_TIME ( yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhh )  
  \STOP_TIME ( yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhh )  
)
  \SCRATCH ( NO | YES )}
);

DELETE EXTERNAL_VENDOR (name32, *, ...)
* delete one or more external vendor definitions
;

DELETE INTERNAL_READER (name32, *, ...)
* delete one or more internal reader definitions
;

DELETE JOB_JCL (name32, *, ...)
* delete one or more job JCL definitions
;

DELETE JOB_SET (name32, *, ...)
* delete one or more job set definitions
;

DELETE JOURNAL_MODEL (name32, *, ...)
* delete one or more journal model definitions
;

DELETE LOG_OF_LOGS (lsn26, *, ...)
* delete repository information for one or more log of logs
;

DELETE LOG_STREAM (lsn26)
* delete repository information for a log stream
;

DELETE OUTPUT_MODEL (name32, *, ...)
* delete one or more output model definitions
;

DELETE RULE_SET (name32, *, ...)
* delete one or more rule set definitions
;

DELETE SEQ_FILE (dsn44, *, ...)
* delete repository information for one or more sequential files
;

DELETE SEQ_GROUP (name32, *, ...)
* delete repository information for one or more sequential file records
* for members of the group
;

DELETE SEQ_GROUP_DEFINITION (name32, *, ...)
* delete one or more sequential group definitions
;
DELETE USER_VARIABLES ( name32, *, ... )
* delete one or more user variable definitions
;

DELETE VSAM_FILE ( dsn44, *, ... )
* delete repository information for one or more VSAM files
;

DELETE VSAM_GROUP ( name32, *, ... )
* delete repository information for one or more VSAM_FILE records
* for members of the group
;

DELETE VSAM_GROUP_DEFINITION ( name32, *, ... )
* delete one or more VSAM group definitions
;

DELETE VSAM_NAME_MODEL ( name32, *, ... )
* delete one or more VSAM name model definitions
;

DELETE VSAM_SET ( name32, *, ... )
* delete one or more VSAM set definitions
;

ENTER_IDCAMS
* execute IDCAMS statements
;

statements to be passed to IDCAMS
EXIT_IDCAMS;

PRINT ARCHIVE_IN ( ddn8_or_dsn44 )
* print an archive
  { SELECTION_EXIT ( program8 ) }
  { COUNT ( from , to ) }
  { START_TIME ( yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhhhhhhhhhhhh ) }
  { STOP_TIME ( yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhhhhhhhhhhhh ) }
;

PRINT LOG_STREAM_IN ( lsn26 { ... } )
* print a log stream
  { VIEW( ACTIVE | ALL ) }
  { DELETE ( NO | YES ) }
  { START_TIME ( yyyyjjjhhmmsst | Xhhhhhhhhhhhhhhhhhhhhhhhhhh ) }
;

PRINT REPOSITORY
* print repository information
  { COUNT ( from , to ) }
;

PRINT UNIT_OF_WORK_IN
* print unit of work information
  { SELECTION_EXIT ( program8 ) }
  { COUNT ( from , to ) }
;
RUVMZ0 Commands

Chapter E Reference Summary

PURGE REPOSITORY
* purge repository records of sequential files
  { PURGE_OUT ( ddn8 ) }
  { SEQ_FILE ( *, ... ) }
  { SEQ_GROUP ( name32, *, ... ) }
  { SCRATCH ( NO | YES ) }
  { SIMULATE }

PURGE REPOSITORY
* purge repository records of VSAM files
  { PURGE_OUT ( ddn8 ) }
  { VSAM_FILE ( *, ... ) }
  { VSAM_GROUP ( name32, *, ... ) }
  { SCRATCH ( NO | YES ) }
  { SIMULATE }

RECOVER BACKOUT
* perform a backout recovery of one or more VSAM files, one or more
* VSAM groups, or both VSAM files and VSAM groups
  { INVENTORY_ONLY (NO | YES ) }
  { RECOVERY_MODE(NIS | SIS) }          <-- global for LSR POOL, etc.
  { BUFND(nnn) }
  { BUFNI(nnn) }
  { LSR_POOL(NO | YES ) }
  VSAM_FILE ( dsn44
    { START_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) }  
    { STOP_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) }
    { SELECTION_EXIT ( program8 ) }
    { RECOVERY_MODE(NIS | SIS) }          <-- local for LSR POOL, etc.
    { BUFND(nnn) }
    { BUFNI(nnn) }
    { LSR_POOL(NO | YES ) }
  )
  { VSAM_FILE ( ... ) }
  VSAM_GROUP ( name32, *, ...  
    { START_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) }  
    { STOP_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) }
    { SELECTION_EXIT ( program8 ) }
    { RECOVERY_MODE(NIS | SIS) }          <-- local for LSR POOL, etc.
    { BUFND(nnn) }
    { BUFNI(nnn) }
    { LSR_POOL(NO | YES ) }
  )
  { VSAM_GROUP ( ... ) }
  { ARCHIVE_IN ( ddn8_or_dsn44, ... ) }

;
RECOVER COMPLETED_UNIT_OF_WORK
* perform a unit-of-work recovery of one or more VSAM files, one or more
* VSAM groups, or both VSAM files and VSAM groups
  \{ INVENTORY ONLY ( NO | YES ) \}
  \{ START TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhh ) \}
STOP_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhh )
VSAM_FILE ( dsn44
  \{ NEW_NAME ( dsn44 ) \}
  \{ START TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhh ) \}
  \{ BACKUP IN ( CURRENT_BACKUP ) | ( ddn8_or_dsn44 ) \}
  \{ SELECTION_EXIT ( program8 ) \}
  )
  \{ VSAM_FILE ( ... ) \}
VSAM_GROUP ( name32, *, …
  \{ START TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhh ) \}
  \{ BACKUP IN ( CURRENT_BACKUP ) \}
  \{ SELECTION_EXIT ( program8 ) \}
  \{ VSAM_NAME_MODEL ( name32 ) \}
  )
  \{ VSAM_GROUP ( ... ) \}
UNIT_OF_WORK_OUT ( ddn8 )
UNIT_OF_WORK_IN ( ddn8_or_dsn44 { , ddn8_or_dsn44, ... } )
ARCHIVE_IN ( ddn8_or_dsn44 { , ddn8_or_dsn44, ... } )
RECOVER FORWARD
* perform a forward recovery of one or more VSAM files, one or more
* VSAM groups, or both VSAM files and VSAM groups
  { INVENTORY_ONLY ( NO | YES ) }
  {RECOVERY_MODE(NIS | SIS) } <-- global for LSR POOL, etc.
  [BUFND(nnn)]
  [BUFNI(nnn)]
  [LSR_POOL(NO | YES )]
  VSAM_FILE ( dsn44 )
    { NEW_NAME ( dsn44 )}
    { START_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhhhh )}
    { STOP_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhhhh )}
    { BACKUP_IN ( CURRENT_BACKUP ) | ( ddn8_or_dsn44 )}
    { SELECTION_EXIT ( program8 )}
    {RECOVERY_MODE(NIS | SIS) } <-- local for LSR POOL, etc.
    [BUFND(nnn)]
    [BUFNI(nnn)]
    [LSR_POOL(NO | YES )]
  )
  [ VSAM_FILE ( ... ) ]
  VSAM_GROUP ( name32, *, ...)
    { START_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhhhh )}
    { STOP_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhhhh )}
    { BACKUP_IN ( CURRENT_BACKUP )}
    { SELECTION_EXIT ( program8 )}
    [ VSAM_NAME_MODEL ( name32 )]
    {RECOVERY_MODE(NIS | SIS) } <-- local for LSR POOL, etc.
    [BUFND(nnn)]
    [BUFNI(nnn)]
    [LSR_POOL(NO | YES )]
  )
  [ VSAM_GROUP ( ... ) ]
  [ ARCHIVE_IN ( ddn8_or_dsn44, ... ) ]

RECOVER REPOSITORY FORWARD
* recover the repository from a backup
* if you omit the merge keyword, all records in the repository
* are deleted before the restore
  { BACKUP_IN ( ddn_dsn )}
  { SMF_IN ( ddn_dsn, ... )}
  { ARCHIVE_IN ( ddn_dsn_list )}
  { START_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhhhh )}
  { STOP_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhhhhhh )}
  { MERGE ( REPLACE | NOREPLACE )}
  { SIMULATE }

REGISTER ARCHIVE_IN ( ddn8_or_dsn44 )
* register an archive file
  { COMMENT ( comment_fields )}
  { LOCATION ( location_name8 )}
  { STATUS ( ACTIVE | INACTIVE )}
### REGISTER BACKUP_FILE (ddn8_or_dsn44)
* register a non-RUV/SMS backup file
  VSAM_FILE (ddn8_or_dsn44) | SEQ_FILE (ddn8_or_dsn44)
  { COMMENT (comment_fields) }
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhh) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }

### REGISTER BACKUP_IN (ddn8_or_dsn44)
* register an RUV backup file
  { COMMENT (comment_fields) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }

### REGISTER LOG_OF_LOGS (lsn26, *, ...)
* register a log of logs
  { DAYS (0 | 45 | nnn | 999) }
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhh) }
  { DELETE (NO | YES) }

### REGISTER RESTORE_JOB_JCL (name32)
* register a backup with the name of the job JCL record to use for restoring the backup
  VSAM_FILE (ddn8_or_dsn44)
  { PARAMETER (userfield64) }
  { COMMENT (comment_fields) }
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhh) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }

### REGISTER RESTORE_METHOD (name32)
* register a backup with the name of the restore method to be reported in the WTOR
* that RUV issues when the backup needs to be restored
  VSAM_FILE (ddn8_or_dsn44)
  { PARAMETER (userfield64) }
  { COMMENT (comment_fields) }
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhhhh) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }

### REORGANIZE VSAM_GROUP (name32)
* make a backup of a group with VSAM data sets and then restore it
  { INVENTORY_ONLY (NO | YES) }
  OUTPUT_MODEL (name32)
  { COMMENT (comment_fields) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }
  { AIX (YES | NO) }
  { RETAIN_BACKUP (NO | YES) }

REORGANIZE VSAM_GROUP (name32)
* make a backup of a group with VSAM data sets with SNAPSHOT and then restore it
  { INVENTORY_ONLY (NO | YES) }
  VSAM_NAME_MODEL (name32)
  { COMMENT (comment_fields) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }

REORGANIZE VSAM_IN (ddn8 or dsn44)
* make a backup of a VSAM data set and then restore it
  { INVENTORY_ONLY (NO | YES) }
  BACKUP_OUT (ddn8)
   { COMMENT (comment_fields) }
   { LOCATION (location_name8) }
   { STATUS (ACTIVE | INACTIVE) }

REORGANIZE VSAM_IN (ddn8 or dsn44)
* make a backup of a group with VSAM data sets with SNAPSHOT and then restore it
  { INVENTORY_ONLY (NO | YES) }
  VSAM_NAME_MODEL (name32)
  { COMMENT (comment_fields) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }

REPORT
* report information about one or more rule set items
  { JOB_JCL (name32, *, ...) }
  { JOB_SET (name32, *, ...) }
  { RULE_SET (name32, *, ...) }
  { VSAM_SET (name32, *, ...) }
  { EXTERNAL_VENDOR (name32, *, ...) }
  { JOURNAL_MODEL (name32, *, ...) }
  { INTERNAL_READER (name32, *, ...) }
  { USER_VARIABLES (name32, *, ...) }
  { OUTPUT_MODEL (name32, *, ...) }
  { SYSTEM_VARIABLES }
  { VSAM_NAME_MODEL (name32, *, ...) }
  { XBMID (ssid, ssid, ...) }

REPORT APPLID (applid8, *, ...)
* report information about one or more CICS APPLIDs
;
REPORT ARCHIVE_FILE (dsn44, *, ...)  
* report information about one or more archive files 
  { FILTER_BY ( 
      { LOCATION (location_name8) } 
      { STATUS (ACTIVE | INACTIVE) } 
      * if you omit FILTER_BY, default is to select both active 
      * and inactive statuses 
      { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) } 
      { STOP_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) } 
  ) } 
;

REPORT AUDIT  
* report on selected archive records 
  { VSAM_FILE ( ... ) } 
  { VSAM_GROUP (name32, *, ...) } 
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) } 
  { STOP_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) } 
  { SELECTION_EXIT (program8) } 
  { TERMID (termid4, *, ...) } 
  { TRANID (tranid4, *, ...) } 
  { SORT_BY (TRANID,TERMID,TIME,VSAM_FILE,KEY,CHRONO) } 
  { ARCHIVE_IN (ddn8_or_dsn44, ...) } 
;

REPORT BACKUP_FILE (dsn44, *, ...)  
* report information about one or more backup files 
  { FILTER_BY ( 
      { LOCATION (location_name8) } 
      { STATUS (ACTIVE | INACTIVE) } 
      * if you omit FILTER_BY, default is to select both active 
      * and inactive statuses 
      { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) } 
      { STOP_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) } 
  ) } 
;

REPORT BACKUP_METHOD ( * | RUV BWO SMS SNAPSHOT EXTERNAL )  
* report information about the backup methods that are enabled for use 
;

REPORT DEFAULT  
* report information about default values 
;

REPORT LOG_OF_LOGS (lsn26, *, ...)  
* report information about one or more log of logs 
;

REPORT LOG_STREAM (lsn26, *, ...)  
* report information about one or more log streams 
;

REPORT PTF  
* report information about applied PTFs 
  { SUBSYSTEM (xxxx) } 
;
REPORT SEQ_FILE (dsn44, *, ...)  
* report information about one or more sequential files

REPORT SEQ_GROUP (name32, *, ...)  
* report information about one or more sequential groups

REPORT SEQ_GROUP_DEFINITION (name32, *, ...)  
* report information about one or more sequential group definitions

REPORT UNIT_OF_WORK (ddn8_or_dsn44, *, ...)  
* report information about one or more unit-of-work files

REPORT VSAM_FILE (dsn44, *, ...)  
* report information about one or more VSAM files

REPORT VSAM_GROUP (name32, *, ...)  
* report information about one or more VSAM groups

REPORT VSAM_GROUP_DEFINITION (name32, *, ...)  
* report information about one or more VSAM group definitions

RESET DEFAULT  
* use internally-defined default values for RUV options  
* instead of user-defined default values

RESTORE REPOSITORY  
* restore a repository data set from a repository backup (BACKUP_IN)  
* that was created with the BACKUP REPOSITORY command
  BACKUP_IN (ddn8_or_dsn44)  
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhh ) }  
  { STOP_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhh ) }  
  { MERGE { ( REPLACE | NOREPLACE ) } }  
* if you omit the MERGE keyword, all records in the repository  
* are deleted before the restore  
  { SIMULATE } 

RESTORE SEQ_FILE (dsn44)  
* restore a sequential file
  { NEW_NAME (dsn44) }  
  { BACKUP_IN (CURRENT_BACKUP) | (dsn44) }  
  { INVENTORY_ONLY (NO | YES) }  
  { SEQ_FILE (...)}
RESTORE SEQ_GROUP (name32
* restore a sequential group
    [ SEQ_NAME_MODEL (name32) ]
    )
    [ INVENTORY_ONLY (NO | YES) ]
    [ SEQ_GROUP (…) ]
;

RESTORE VSAM_FILE (dsn44
* restore a VSAM file
    [ NEW_NAME (dsn44) ]
    [ AIX (YES | NO) ]
    [ BACKUP_IN (CURRENT_BACKUP) | (dsn44) ]
    )
    [SNAP_BACK(YES | NO) ]  <-- for REORGANIZE
    [ INVENTORY_ONLY (NO | YES) ]
    [ VSAM_FILE (…) ]
;

RESTORE VSAM_GROUP (name32
* restore a VSAM group
    [ VSAM_NAME_MODEL (name32) ]
    [ AIX (YES | NO) ]
    )
    [ INVENTORY_ONLY (NO | YES) ]
    [SNAP_BACK(YES | NO) ]  <-- for REORGANIZE
    [ VSAM_GROUP (…) ]
;
SET
* set values for RUV options
  { ASSOCIATED_ADDRESS_SPACE_TASKS ( 0 | 10 | nnn | 200 ) }
  { BACKUP_CHANGED_ONLY ( NO | YES ) }
  { BACKUP_METHOD(
    RUV
    { BACKUP_REPLY }
    { BWO | NO_BWO } 
    { SNAPSHOT | NO_SNAPSHOT }
    { SMS | NO_SMS } 
    { EXTERNAL | NO_EXTERNAL } 
  )}
  { DATE_FORMAT ( YYYY.JJJ | MM/DD/YYYY | DD/MM/YYYY | YYYY/MM/DD | USER-DEFINED )}
  { DAYS ( 0 | 45 | nnn | 999 ) | CYCLES ( 0 | nnn | 999 )}
  { ESOTERIC (device) }
  { GAP ( STOP | WARN | IGNORE )}
  { INCREMENTAL_BACKUP_SUPPORT ( YES | NO )}
  { LANGUAGE ( ENU )}
  { LINE_COUNT ( 15 | 56 | nnnnnn | 99999999 )}
  { MAX_CC ( nn )}
  { PAGE_WIDTH ( 30 | nnn | 121 )}
  { PASSWORD_EXPIRATION_CC ( 0 | 4 )}
  { <+1> ( 1 | nnnnnn | 99999999 )}
  { READER_TASKS ( 1 | nnn | 100 )}
  { REPORT ( SUMMARY | DETAIL | FULL )}
  { RLS_RECOVER_PROTECTION ( NO | YES )}
  { SMF_TYPE ( x,y )}
* SMF_TYPE x valid values are 0 and 128–255
* SMF_TYPE y valid values are 0 and 1–10
  { SUBSYSTEM (xxxx )}
  { SYNTAX_CHECK ( NO | YES )}
  { TERM_CC ( n1, n2 )}
* TERM_CC default value is 4, 0
  { TIME_FORMAT ( MH:MM:SS | HH:MM:SS_A/PM )}
  { UPPER_CASE_TRANSLATION ( OFF | ON )}
  { USE_AUTOJOURNAL ( NO | YES )}
  { WORK_DISK ( devicename8 )}
  { WORK_TAPE ( devicename8 )}
  { XBID ( ssid1 ssid2 ssid3 ... )}
;
SET DEFAULT
* set default values for RUV options
  { ASSOCIATED_ADDRESS_SPACE_TASKS (0 | 10 | nnn | 200 ) }
  { BACKUP_CHANGED_ONLY (NO | YES) }
  { BACKUP_METHOD( 
    RUV 
    { BACKUP_REPLY } 
    { BWQ | NO_BWO } 
    { SNAPSHOT | NO_SNAPSHOT } 
    { SMS | NO_SMS } 
    { EXTERNAL | NOEXTERNAL } 
  ) } 
  { DATE_FORMAT ( 
    YYYYJJJ | 
    MM/DD/YYYY | 
    DD/MM/YYYY | 
    YYYY/MM/DD | 
    USER-DEFINED 
  ) }
  { DAYS (0 | 45 | nnn | 999) | CYCLES (0 | nnn | 999) }
  { ESOTERIC (device) }
  { GAP (STOP | WARN | IGNORE) }
  { INCREMENTAL_BACKUP_SUPPORT (YES | NO) }
  { LANGUAGE (ENU) }
  { LINE_COUNT (15 | 56 | nnnnnnn | 99999999) }
  { MAX_CC (nn) }
  { PAGE_WIDTH (30 | nnn | 121) }
  { PASSWORD_EXPIRATION_CC (0 | 4) }
  { <+1> (1 | nnnnnnn | 99999999) }
  { READER_TASKS (1 | nnn | 100) }
  { RECOVERY_MODE (NIS) }
  { RLS_RECOVER_PROTECTION (NO | YES) }
  { SMF_TYPE (x,y) }
* SMF_TYPE x valid values are 0 and 128–255
* SMF_TYPE y valid values are 0 and 1–10
  { SUBSYSTEM (xxxx) }
  { SYNTAX_CHECK (NO | YES) }
  { TERM_CC (n1, n2) }
* TERM_CC default value is 4, 0
  { TIME_FORMAT (MH:MM:SS | HH:MM:SS_A/PM) }
  { UPPER_CASE_TRANSLATION (OFF | ON) }
  { USE_AUTOJOURNAL (NO | YES) }
  { WORK_DISK (devicename8) }
  { WORK_TAPE (devicename8) }
  { XBMID (ssid1 ssid2 ssid3 ...) }
)

STORE INTO_FILE (dsn44_or_pdsmember | <variables>);
* store information into a file for later retrieval
<set_of_variables>
<more_variables>
END_DATA;
SUBMIT BACKUP
* submit a backup job JCL record for execution
  JOB_JCL ( name32 )
  { VSAM_GROUP ( name32, ... ) | VSAM_FILE ( ddn_or_dsn )
   INTERNAL_READER ( name32 )
   INVENTORY_ONLY ( NO | YES )
  ;

TEST DELAY(ss)
* specify the length of time, in seconds, for RUV to pause before continuing with execution

TRY
* test a rule set in a batch environment
  JOB ( jobname8 )
  PROGRAM ( program8 )
  DSN ( dsn44 )

UPDATE ARCHIVE_FILE ( ddn8_or_dsn44 )
* update repository information for the indicated archive file
  { REGISTRATION_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhh ) }
  { COMMENT ( comment_fields ) }
  { LOCATION ( location_name8 ) }
  { STATUS ( ACTIVE | INACTIVE ) }

UPDATE ARCHIVE_FILE ( ddn8_or_dsn44,*,... )
* update repository information for one or more archive files
  { FILTER_BY (}
    { LOCATION ( location_name8 ) }
    { STATUS ( ACTIVE | INACTIVE ) }
  * if you omit FILTER_BY, default is to update both active
  * and inactive statuses
    { START_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhh ) }
    { STOP_TIME ( yyyyjjjhhmmssst | Xhhhhhhhhhhhhhh ) }
  )
  { COMMENT ( comment_fields ) }
  { LOCATION ( location_name8 ) }
  { STATUS ( ACTIVE | INACTIVE ) }


UPDATE BACKUP_FILE (ddn8_or_dsn44)
* update repository information for a backup file
  { REGISTRATION_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) }
  { COMMENT (comment_fields) }
  { LOCATION (location_name8) }
  { STATUS (ACTIVE | INACTIVE) }
  { FILTER_BY (}
    { LOCATION (location_name8) }
    { STATUS (ACTIVE | INACTIVE) }
    * if you omit FILTER_BY, default is to update both active
    * and inactive statuses
    { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) }
    { STOP_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhhh ) }
  ) }

UPDATE EXTERNAL_VENDOR (name32, *, ...)
* update an external vendor definition
  { COMMENT (comment_fields) }
  { BACKUP_REGISTRATION ( NO | program8
    ["initialization parameters"
  ) }
  { JOURNAL_REGISTRATION ( NO | program8
    ["initialization parameters"
  ) }
  { STATUS (ACTIVE | INACTIVE) }

UPDATE INTERNAL_READER (name32)
* update an internal reader definition to store JCL
* in the indicated PDS member
  { COMMENT (comment_fields) }
  PDS_OUT (pds.dsn44.with.variables)
  MEMBER (var_permitted)

UPDATE INTERNAL_READER (name32)
* update an internal reader definition to write JCL
* to the indicated JES job class
  { COMMENT (comment_fields) }
  CLASS (a)

UPDATE INTERNAL_READER (name32)
* update an internal reader definition to store JCL
* in a data set that has the indicated ddname
  { COMMENT (comment_fields) }
  DDNAME (ddn8)
UPDATE JOB_JCL (name32)
* update a job JCL definition
  
  { COMMENT (comment_fields) } {
;
//<JOBNAME(1,8)> JOB (<ACCT>),CLASS=x, ...
//RUZSM0 EXEC PGM=RUZSM0
<RUV_STEPLIB>
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
<RUV_CONTROL>
<RUV_JOURNAL_DEACTIVATE>
END_DATA;

UPDATE JOB_SET
* update a job set definition
  
  JOB_RULE (job_mask8,program_mask8),
  
  { COMMENT (comment_fields) }
;

UPDATE JOURNAL_MODEL (name32)
* update a journal model definition with QSAM statements
  
  { COMMENT (comment_fields) }
  
  { PROTECTION_LEVEL (1 | 0) }
* QSAM statement coded as in JCL
* RUV variables are permitted
  
  QSAM (" DSN=xxxx.xxxxx.xxxx,
        " SPACE=(xxx,(xxx,xxx)),
        " UNIT=xxxx, DISP=(xxx )")
;

UPDATE JOURNAL_MODEL (name32)
* update a journal model definition as a data set
  
  { COMMENT (comment_fields) }
  
  { PROTECTION_LEVEL (1 | 0) }
  
  DDNAME (ddn8)
;

UPDATE LOG_OF_LOGS (lsn26)
* update repository information for the indicated log of logs
  
  { COMMENT (comment_fields) }
  
  { STATUS (ACTIVE | INACTIVE) }
  
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhhh) }
;

UPDATE LOG_STREAM (lsn26)
* update repository information for the indicated log stream
  
  { COMMENT (comment_fields) }
  
  { STATUS (ACTIVE | INACTIVE) }
  
  { START_TIME (yyyyjjjhhmmssst | Xhhhhhhhhhhhhhh) }
;
UPDATE OUTPUT_MODEL (name32)
* update an output model definition
  { COMMENT (comment_fields) }
  { BACKUP_OUT_TYPE(DASD | TAPE | STACKED_TAPE) }
  { STACKED_TAPE_COUNT(1 | nn) }
* QSAM statement as in JCL
* RUV variables are permitted
  QSAM (" DSN=dsssn.xxxx.xxx, "
  " SPACE=(xxx,xxx,xxx), "
  " UNIT=xxxx, DISP=(xxx )")
;
UPDATE RULE_SET (name32)
* update a rule set definition
  { COMMENT (comment_fields) }
  JOB_SET (name32, ...)
  VSAM_SET (name32, ...)
  USER_VARIABLES (name32, ...)
  EXTERNAL_VENDOR (name32, ...)
;
UPDATE SEQ_FILE (dsn44)
* update repository information for the indicated sequential file
  { COMMENT (comment_fields) }
  { DAYS( nnn ) | CYCLES( nnn) }
  { SELECTION_EXIT (name8) }
;
UPDATE SEQGROUP (name32)
* update repository information for the indicated sequential group
  { COMMENT (comment_fields) }
  { DAYS( nnn ) | CYCLES( nnn) }
  { SELECTION_EXIT (name8) }
;
UPDATE SEQGROUP_DEFINITION (name32)
* update a sequential group definition
  { COMMENT (comment_fields) }
  SEQ_FILE (dsn_mask44,
    { INCLUDE | EXCLUDE }
  )
  SEQ_FILE ( ...)
;
UPDATE VSAM_FILE (dsn44)
* update repository information for the indicated VSAM file
  { COMMENT (comment_fields) }
  { APPLID_FILEID (applid8, fileid8) }
  { APPLID_FILEID ( ... ) }
  { DAYS( nnn ) | CYCLES( nnn) }
  { SELECTION_EXIT (name8) }
;
<table>
<thead>
<tr>
<th>Command Name</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE VSAM_GROUP (name32)</td>
<td>* update repository information for the indicated VSAM group</td>
<td><code>UPDATE VSAM_GROUP (name32)</code> <code>* update repository information for the indicated VSAM group</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>{ COMMENT (comment_fields) }</code> `{ DAYS (nnn)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>;</code></td>
</tr>
<tr>
<td>UPDATE VSAM_GROUP_DEFINITION (name32)</td>
<td>* update a VSAM group definition</td>
<td><code>UPDATE VSAM_GROUP_DEFINITION (name32)</code> <code>* update a VSAM group definition</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>{ COMMENT (comment_fields) }</code> VSAM_FILE (<code>dsn_mask44</code>), `{ INCLUDE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VSAM_FILE (<code>...</code>)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>;</code></td>
</tr>
<tr>
<td>UPDATE VSAM_NAME_MODEL (name32)</td>
<td>* update a VSAM name model definition</td>
<td><code>UPDATE VSAM_NAME_MODEL (name32)</code> <code>* update a VSAM name model definition</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>{ COMMENT (comment_fields) }</code> `{ REPLACE (NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`{ STATUS (ACTIVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>;</code></td>
</tr>
<tr>
<td>UPDATE USER_VARIABLES (name32)</td>
<td>* update a user variables definition</td>
<td><code>UPDATE USER_VARIABLES (name32)</code> <code>* update a user variables definition</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>{ COMMENT (comment_fields) }</code> <code>&lt;variablename&gt; /variablevalue/</code> END_DATA;`</td>
</tr>
<tr>
<td>VALIDATE REPOSITORY</td>
<td>* detect and correct repository errors</td>
<td><code>VALIDATE REPOSITORY</code> <code>* detect and correct repository errors</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>{ SIMULATE }</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>;</code></td>
</tr>
<tr>
<td>VALIDATE SUBSYSTEM (ssid)</td>
<td>* perform a subsystem validation</td>
<td><code>VALIDATE SUBSYSTEM (ssid)</code> <code>* perform a subsystem validation</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>`{ DUMP (NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>;</code></td>
</tr>
</tbody>
</table>
User Variables

You can use user-defined variables, RUV-defined variables, and IBM-defined variable substitution symbols when you create journal model, skeleton JCL, internal reader, and VSAM name model entries.

User-Defined Variables

Use the ADD USER_VARIABLES command to define your own user-defined variables. Define a user-defined variable by coding a variable name, which begins with the less-than character (<) and ends with the greater-than character (>), and its variable substitution value, which can contain other system or user-defined variables. Each user-defined variable is limited to one command line (71 characters). Indicate the end of the list of user-defined variables by coding the END_DATA keyword. For more information, see page 688.

Substring Notation

You can select portions of a user variable by using substring notation, as shown in the following example:

\(<\text{USER_VARIABLE \{ (start\_location \{, length \}) \} } >\)

The start_location and length values indicate position and count. The default start_location number is 1. If the start_location number is larger than the length of the variable, RUV substitutes a null value. Use a minus sign in the start_location value to specify a count from the end of the variable and a move right by the length value. If you do not provide a length value, RUV assumes that you want to use the remainder of the string. The following table shows examples with the job name PRODGNL.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Result</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;PRODGNL&gt;</td>
<td>PRODGNL</td>
<td>default uses all characters</td>
</tr>
<tr>
<td>&lt;PRODGNL(3,2)&gt;</td>
<td>OD</td>
<td>start at third character, count 2</td>
</tr>
<tr>
<td>&lt;PRODGNL(2)&gt;</td>
<td>PR</td>
<td>start at first character, count 2</td>
</tr>
<tr>
<td>&lt;PRODGNL(5)&gt;</td>
<td>GNL</td>
<td>start at fifth character</td>
</tr>
<tr>
<td>&lt;PRODGNL(9,3)&gt;</td>
<td></td>
<td>9 too large, null substitution</td>
</tr>
<tr>
<td>&lt;PRODGNL(-3)&gt;</td>
<td>GNL</td>
<td>go to end, move back 3</td>
</tr>
</tbody>
</table>
RUV-Defined Variables

RUV provides the following predefined variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BACKUP_DSN&gt;</td>
<td>The data set name of the backup data set. This variable is the value of the BACKUP_FILE keyword (1–44 characters) from the REGISTER BACKUP_FILE command that was used to register the backup. This variable is valid in restore job JCL records.</td>
</tr>
<tr>
<td>&lt;BACKUP_TIME&gt;</td>
<td>The start time, in yyyyjjjjhhmmss format, of the backup. This variable is the value of the START_TIME keyword. This variable is valid in restore job JCL records.</td>
</tr>
<tr>
<td>&lt;JOBNAME&gt;</td>
<td>Standard z/OS JCL job statement name that executes an initiator.</td>
</tr>
<tr>
<td>&lt;JOB_NUMBER&gt;</td>
<td>Standard z/OS JES job number.</td>
</tr>
<tr>
<td>&lt;LOOP_TIME&gt;</td>
<td>A time, in yyyyjjjjhhmmss format, that RUV assigns to this variable when it encounters the &lt;:BEGIN_LOOP&gt; action variable. This time value does not change until RUV finishes processing the members of the group and exits the loop.</td>
</tr>
<tr>
<td>&lt;LOOP_TIME_HEX&gt;</td>
<td>A time, in Xhhhhhhhhhhhhhhhh format, that RUV assigns to this variable when it encounters the &lt;:BEGIN_LOOP&gt; action variable. This time value does not change until RUV finishes processing the members of the group and exits the loop.</td>
</tr>
<tr>
<td>&lt;NODEn&gt;</td>
<td>The contents of the node (qualifier) in the input data set name at the position that is indicated by n; for example, if the input data set name is PAYROLL.TAXWTHLD.DATA, the variable &lt;NODE2&gt; refers to the string TAXWTHLD.</td>
</tr>
<tr>
<td>&lt;PARAMETER&gt;</td>
<td>A 64-byte user-defined field. The value is provided at registration time. This variable is valid in backup and restore job JCL records.</td>
</tr>
<tr>
<td>&lt;PGMNAME&gt;</td>
<td>Standard z/OS JCL EXEC job statement name for program being executed.</td>
</tr>
<tr>
<td>&lt;PROCNAME&gt;</td>
<td>Standard z/OS JCL procedure name in a PROCLIB that will produce a set of JCL.</td>
</tr>
<tr>
<td>&lt;RUVBACKUP&gt;</td>
<td>consolidates &lt;RUVDATE&gt; and &lt;RUVTIME&gt; into one variable.</td>
</tr>
<tr>
<td>&lt;RUV_CONTROL&gt;</td>
<td>A place-holder specification for all RUV commands and keywords. Used in JCL and ISPF. This variable indicates the position in which all control statements will be placed in your job.</td>
</tr>
<tr>
<td>&lt;RUVDATE&gt;</td>
<td>The current Julian date Dyyyyjjj: 4-digit year, 3-digit day of the year (1 to 366).</td>
</tr>
<tr>
<td>&lt;RUVTIME&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;RUV_JOB_JCL&gt;</td>
<td>The 32-character name of the backup job JCL record that was specified in the SUBMIT command.</td>
</tr>
</tbody>
</table>
RUV-Defined Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;RUV_JOURNAL&gt;</code></td>
<td>The journal data set name (44-characters) that is active for the batch journaling facility. Whenever a batch job is running and it has journaling, a user-set file will be opened named RUV_JOURNAL. Used primarily for backout situations.</td>
</tr>
<tr>
<td><code>&lt;RUV_JOURNAL_DEACTIVATE&gt;</code></td>
<td>Used to deactivate logs and journals after a backout. Used in JOB_JCL models; valid only in backout jobs. Causes the following statement to be added for each log or journal created in a job step:</td>
</tr>
<tr>
<td></td>
<td>UPDATE ARCHIVE_FILE(archive_dsn) REGISTRATION_TIME(Xhhhhhhhhhhhhhhhhhh) STATUS(INACTIVE);</td>
</tr>
<tr>
<td></td>
<td>Xhhhhhhhhhhhhhhhhhh is replaced with the registration time of the archive or archives.</td>
</tr>
<tr>
<td><code>&lt;RUV_LOG&gt;</code></td>
<td>The logging data set name (44-characters) that is active for the batch journaling facility. Whenever a batch job is running and it has before-image logging, a user-set file will be opened named RUV_LOG. Used primarily for backout situations.</td>
</tr>
<tr>
<td><code>&lt;RUV_STEPLIB&gt;</code></td>
<td>Propagates the current z/OS load libraries and RUVEVLIB to the recovery job. Used in JOB_JCL models. Enter this variable on a line by itself. It might produce multiple lines of output.</td>
</tr>
<tr>
<td><code>&lt;RUVTIME&gt;</code></td>
<td>The current GMT time in 24-hour military time: Thhmmssst 2-digit hour (00 to 23), 2-digit minute (00 to 59), 2-digit second (00 to 59), and 1-digit tenths of a second (0 to 9).</td>
</tr>
<tr>
<td><code>&lt;RUVTIME_HEX&gt;</code></td>
<td>The current GMT time in hexadecimal format: Xhhhhhhhhhhhhhhhhhh.</td>
</tr>
<tr>
<td><code>&lt;SMFID&gt;</code></td>
<td>Standard z/OS JCL 4-character name that identifies a system (like SYSx).</td>
</tr>
<tr>
<td><code>&lt;STEPNAME&gt;</code></td>
<td>Standard z/OS JCL job step name that executes in an initiator.</td>
</tr>
<tr>
<td><code>&lt;SYSNDX&gt;</code></td>
<td>A seven-digit number that works with <code>&lt;BEGIN_LOOP&gt;</code> and <code>&lt;END_LOOP&gt;</code> action variables that define a processing loop. <code>&lt;BEGIN_LOOP&gt;</code> sets the value of <code>&lt;SYSNDX&gt;</code> to 0000001. RUV processes the statements within the loop for each member of a VSAM group and increments the value of <code>&lt;SYSNDX&gt;</code> by one for each loop.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SYSNDX&gt;</code> is valid in backup and restore job JCL records.</td>
</tr>
<tr>
<td><code>&lt;USERID&gt;</code></td>
<td>Standard z/OS JCL job statement value that identifies the user.</td>
</tr>
<tr>
<td><code>&lt;VSAM_DEFINITION&gt;</code></td>
<td>Build IDCAMS Define function statements for a VSAM file. This variable is valid only within a set of <code>&lt;BEGIN_LOOP&gt;</code> and <code>&lt;END_LOOP&gt;</code> action variables.</td>
</tr>
<tr>
<td><code>&lt;VSAM_PATH_DEFINITION&gt;</code></td>
<td>Build IDCAMS Define function statements for all associated alternate indexes and paths. BLDINDEX commands are also created. This variable is valid only within a set of <code>&lt;BEGIN_LOOP&gt;</code> and <code>&lt;END_LOOP&gt;</code> action variables.</td>
</tr>
</tbody>
</table>
IBM Variable Substitution Symbols

RUV supports IBM-type variable (such as &XXX.&YYY) substitution, as shown in the following example:

```
GL.&SYSUID.&DATE.T<RUVTIME(2,5)>.
```

You can display the IBM symbols that are defined at your data center by entering the command D SYMBOLS at the z/OS console.

RUV Action Variables

You can use RUV action variables in backup and restore job JCL records to control the variable substitution process. Action variables are enclosed by the less-than symbol and the colon (<:) and a greater-than (>) symbol; <:BEGIN_LOOP> is an example of an action variable. The <: combination is reserved by RUV and is always interpreted as an action variable. Do not use this combination for any other purpose.

<table>
<thead>
<tr>
<th>Action Variable</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;:BEGIN_LOOP&gt;</td>
<td>Code the &lt;:BEGIN_LOOP&gt; action variable at the beginning of a set of JCL statements that you want RUV to process for each member of the specified VSAM group. &lt;:BEGIN_LOOP&gt; also resets the &lt;SYSNDX&gt; variable to 0000001. Loops cannot be nested; close a loop that is started with the &lt;:BEGIN_LOOP&gt; action variable by coding the &lt;:END_LOOP&gt; action variable before you code another &lt;:BEGIN_LOOP&gt; action variable.</td>
</tr>
<tr>
<td>&lt;:END_LOOP&gt;</td>
<td>Code the &lt;:END_LOOP&gt; action variable at the end of a set of JCL statements that you want RUV to process for each member of the specified VSAM group.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;VSAM_DSN &amp;</td>
<td>A user-specified data set name (44-characters). If you specify this variable in a backup or restore job JCL record, it is valid only within a set of &lt;:BEGIN_LOOP&gt; and &lt;:END_LOOP&gt; action variables.</td>
</tr>
<tr>
<td>&amp;+1</td>
<td>This variable allows you to change a value incrementally by adding a number to the end of it. Each time you use this variable, it adds 1 to a hidden number n (a 7-character number formatted as V0000000). The first time the variable is used, it is incremented to 1 (V0000001). The next time it is used, it is 2 (V0000002), and so forth.</td>
</tr>
</tbody>
</table>

- IBM Variable Substitution Symbols
- RUV supports IBM-type variable (such as &XXX.&YYY) substitution, as shown in the following example:

```
GL.&SYSUID.&DATE.T<RUVTIME(2,5)>.
```

You can display the IBM symbols that are defined at your data center by entering the command D SYMBOLS at the z/OS console.

- RUV Action Variables

You can use RUV action variables in backup and restore job JCL records to control the variable substitution process. Action variables are enclosed by the less-than symbol and the colon (<:) and a greater-than (>) symbol; <:BEGIN_LOOP> is an example of an action variable. The <: combination is reserved by RUV and is always interpreted as an action variable. Do not use this combination for any other purpose.

<table>
<thead>
<tr>
<th>Action Variable</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;:BEGIN_LOOP&gt;</td>
<td>Code the &lt;:BEGIN_LOOP&gt; action variable at the beginning of a set of JCL statements that you want RUV to process for each member of the specified VSAM group. &lt;:BEGIN_LOOP&gt; also resets the &lt;SYSNDX&gt; variable to 0000001. Loops cannot be nested; close a loop that is started with the &lt;:BEGIN_LOOP&gt; action variable by coding the &lt;:END_LOOP&gt; action variable before you code another &lt;:BEGIN_LOOP&gt; action variable.</td>
</tr>
<tr>
<td>&lt;:END_LOOP&gt;</td>
<td>Code the &lt;:END_LOOP&gt; action variable at the end of a set of JCL statements that you want RUV to process for each member of the specified VSAM group.</td>
</tr>
</tbody>
</table>
## RUV Action Variables

<table>
<thead>
<tr>
<th>Action Variable</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;:IMBED imbed_source&gt;</code></td>
<td>Code the <code>&lt;:IMBED outside_source&gt;</code> action variable to insert statements from a source that is outside of the job JCL record. The string <code>outside_source</code> represents the ddname or data set name of a flat file or partitioned data set member that contains fixed-length, 80-byte records. After RUV inserts the statements from the outside source, RUV continues to perform variable substitution for any variables. RUV supports nested <code>&lt;:IMBED outside_source&gt;</code> action variables for a maximum of ten levels. You can use variables to build the ddname or data set name that identifies the outside source. Code the <code>&lt;:IMBED outside_source&gt;</code> action variable on a line by itself. You can also use the <code>&lt;:IMBED outside_source&gt;</code> action variable with the STORE INTO_FILE command.</td>
</tr>
<tr>
<td><code>&lt;:VSOFF&gt;</code></td>
<td>Code the <code>&lt;:VSOFF&gt;</code> action variable to disable RUV substitution and echo the input statements as they are coded. You can code <code>&lt;:VSOFF&gt;</code> in any position in the input statement.</td>
</tr>
<tr>
<td><code>&lt;:VSON&gt;</code></td>
<td>Code the <code>&lt;:VSON&gt;</code> action variable to resume RUV substitution after you have disabled it with <code>&lt;:VSOFF&gt;</code>. You can code <code>&lt;:VSOFF&gt;</code> in any position in the input statement.</td>
</tr>
</tbody>
</table>
RUV Subsystem Commands

The following sections describe the commands that you can use to control RUV elements of the BMC Primary Subsystem (BMCP), the BMC Consolidated Subsystem (BCSS), and the Backup-While-Open (BWO) Subsystem (RUVM).

BMCP Console Commands

You can enter the following BMCP commands from the operating system console or a TSO SDSF session.

<table>
<thead>
<tr>
<th>BMCP Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCP STATUS</td>
<td>Display the status of the BMCP.</td>
</tr>
<tr>
<td>BMCP SHUTDOWN</td>
<td>Shut down the BMCP in an orderly manner by terminating the BMCP address space and BMCP command processing.</td>
</tr>
</tbody>
</table>

BCSS Console Commands

You can enter the following BCSS commands from the operating system console or a TSO SDSF session. These commands are applicable to both the public and private copies of the subsystem.

<table>
<thead>
<tr>
<th>BCSS Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssid REINIT RUV</td>
<td>Reinitialize the RUV components of the identified BCSS, and load any modules that are new or changed from the subsystem STEPLIB into CSA.</td>
</tr>
<tr>
<td>ssid RUV DEBUG</td>
<td>Display internal diagnostic information.</td>
</tr>
<tr>
<td></td>
<td>{ BWOMSGL1</td>
</tr>
<tr>
<td>ssid RUV DISABLE</td>
<td>Immediately terminate all current Batch Journaling Facility and BWO processes that are in-flight, and prevent any new processes from being started. If you specify ALL or LOGGING, all batch jobs that are using the Batch Journaling Facility are terminated immediately.</td>
</tr>
<tr>
<td></td>
<td>{ ALL</td>
</tr>
<tr>
<td>ssid RUV ENABLE</td>
<td>Allow the Batch Journaling Facility and BWO processes to start again after a DISABLE or QUIESCE command.</td>
</tr>
<tr>
<td></td>
<td>{ ALL</td>
</tr>
</tbody>
</table>
BWO CICS Commands

You can enter the following BWO CICS (RUVM) commands from the operating system console or a CICS terminal. From the operating system console, start each command with a z/OS MODIFY command. In the following table, cicsjob is the job name of the CICS region.

<table>
<thead>
<tr>
<th>BCSS Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssid RUV HELP</td>
<td>Display a short list of RUV subsystem commands on the z/OS console.</td>
</tr>
<tr>
<td>ssid RUV QUIESCE {ALL</td>
<td>LOGGING</td>
</tr>
<tr>
<td>ssid RUV STATUS {SECURITY}</td>
<td>Display the current status of the Batch Journaling Facility and BWO feature, and display the security level.</td>
</tr>
<tr>
<td>ssid RUV TAKEOVER BWOCOORD</td>
<td>Cause the target subsystem to take over coordination of BWO processes.</td>
</tr>
<tr>
<td>ssid SHUTDOWN</td>
<td>Cause the target RUV subsystem to terminate its processing.</td>
</tr>
<tr>
<td>ssid STATUS</td>
<td>Display the status of the BCSS address space.</td>
</tr>
<tr>
<td>ssid RUV ACTIVATE RULE_SET( name32 )</td>
<td>TRACE_ON</td>
</tr>
</tbody>
</table>

BWO CICS Commands

You can enter the following BWO CICS (RUVM) commands from the operating system console or a CICS terminal. From the operating system console, start each command with a z/OS MODIFY command. In the following table, cicsjob is the job name of the CICS region.

<table>
<thead>
<tr>
<th>RUVM Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F cicsjob,RUV FORCE RUVM FORCE</td>
<td>Terminate all subtasks controlled by the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td>F cicsjob,RUVM MONITOR RUVM MONITOR</td>
<td>Start a CICS task that displays at the CICS terminal the status of all subtasks controlled by the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td>F cicsjob,RUVM RESTART RUVM RESTART</td>
<td>Perform a shutdown and then a start of the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td>F cicsjob,RUVM START RUVM START</td>
<td>Start the Backup control task within CICS that communicates with the RUVM BWO Subsystem.</td>
</tr>
<tr>
<td>F cicsjob,RUVM STATUS RUVM STATUS</td>
<td>Display the status of any active subtasks performing backups for a Batch Backup job.</td>
</tr>
<tr>
<td>F cicsjob,RUVM SHUTDOWN RUVM SHUTDOWN</td>
<td>Shut down the RUVM BWO Subsystem.</td>
</tr>
</tbody>
</table>
User Exit Parameter List

RUUV provides a user exit for the RECOVER BACKOUT, RECOVER FORWARD, and PRINT ARCHIVE_IN commands. You specify the user exit program to execute by specifying the SELECTION_EXIT(program) keyword. The user exit is called at beginning of processing, once for each record, and at the end of the processing.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 parm_length</td>
<td>4</td>
<td>the parameter list length (includes itself)</td>
<td></td>
</tr>
<tr>
<td>02 dsn</td>
<td>4</td>
<td>a pointer to the 44-character data set name to which this record is associated</td>
<td></td>
</tr>
<tr>
<td>03 key length</td>
<td>4</td>
<td>the key length that is associated with the data set name</td>
<td></td>
</tr>
<tr>
<td>04 key</td>
<td>4</td>
<td>a pointer to the key for this record</td>
<td></td>
</tr>
<tr>
<td>05 data length</td>
<td>4</td>
<td>the length of the data record</td>
<td></td>
</tr>
<tr>
<td>06 data record</td>
<td>4</td>
<td>a pointer to the data record</td>
<td></td>
</tr>
<tr>
<td>07 action</td>
<td>4</td>
<td>a pointer to a 2-byte action value:\</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’00 X’00’ — initialize call</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AA — data add, record applies on recovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AP — data add, call from PRINT ARCHIVE_IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA — data add complete, record applies on recovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CP — data add complete, call from PRINT ARCHIVE_IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DP — delete record, call from PRINT ARCHIVE_IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LP — logical unit of work record from PRINT ARCHIVE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RA — data read for update, record applies on recovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RP — data read for update, call from PRINT ARCHIVE_IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UA — data update delete, record applies on recovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UP — data update delete, call from PRINT ARCHIVE_IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X’FF X’00’ — termination call</td>
<td></td>
</tr>
<tr>
<td>08 jobname</td>
<td>4</td>
<td>a pointer to an 8-byte field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If a batch job caused this record, it is an 8-byte jobname. If CICS caused this record, the field is an applid.</td>
<td></td>
</tr>
<tr>
<td>09 timestamp</td>
<td>4</td>
<td>a pointer to an area of three or six contiguous fields:\</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting UOW 8-byte TOD store clock value</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting UOW 4-byte packed date ‘0CYYJJs’</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting UOW 4-byte packed time ‘HHMMSSTs’</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ending UOW 8-byte TOD store clock value</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ending UOW 4-byte packed date ‘0CYYJJs’</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ending UOW 4-byte packed time ‘HHMMSSTs’</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Bytes</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 10        | origination | 4     | a pointer to a 1 byte value that indicates the origin:  
|           |          |       | x'01' — Batch  
|           |          |       | The following values are for CICS only:  
|           |          |       | x'02' — 0.3.3.x RPCV version  
|           |          |       | x'03' — 4.2.x.4 RPCV version  
|           |          |       | x'04' — 5.0.0.0 RPCV version or later  
|           |          |       | x'05' — 0.0.0.1 Log Block version  
|           |          |       | x'06' — 0.0.0.1 CICS journal input  
| 11        | user token | 4     | a user-defined token  
|           |          |       | On the first call to the user exit, this value will be x'00000000'. You may set this value only on the first call. The value you set will be returned on subsequent calls.  
| 12        | task number | 4     | CICS only - a pointer to a 4-byte packed decimal value  
| 13        | TRANID    | 4     | CICS only - a pointer to a 4-byte character value known as the CICS transaction ID  
| 14        | TERMID    | 4     | CICS only - a pointer to a 4-byte character value that is the terminal ID as defined by CICS  

a The first byte identifies the type of action and the second byte identifies the source of the call.  
b Starting or ending UOW fields may be all x'00' to indicate an incomplete unit of work.
Glossary

This glossary contains terminology relating to Recovery Utility for VSAM (RUV). The entries are arranged alphabetically.

A

ACB
See access method control block.

access method control block
ACB. A control block that links an application program to VSAM or ACF/VTAM.

action
One of the defined tasks that an application performs.

action bar
The area at the top of the primary window containing keywords that give you access to actions available in that window.

accumulation
A set of journal records in file ID, key, and date-time sequence. Also referred to as ACCUM processing.

AIX
See alternate index.

after-image
The journal record of a file transaction (add, delete, or update) after the transaction occurs. Contrast with before-image.

alternate index
A collection of index entries related to a given base cluster and organized by an alternate key (a key other than the primary key of the associated base cluster data records). It gives an alternate directory for finding records in the data component of a base cluster. See also path.

APPLICATION RESTART CONTROL for VSAM
A BMC Software product that provides checkpoint/restart support and repositioning of VSAM data sets at application restart. When used in conjunction with RUV, it provides automatic logging and backout of changes to VSAM data sets.
Applid
   An application ID that is defined to CICS for communication purposes.

archiving
   The storage of backup files and any associated journal records, usually for a given period of
time.

AR/CTL™
   The acronym for APPLICATION RESTART CONTROL.

ASID
   A unique system-assigned identifier for an address space.

B

backout
   The process of restoring a previous state of all or part of a system. All updates against protected
resources are removed. Backout occurs during a restart in the event of CICS or z/OS image
failure or a batch job step failure.

back up
   To copy information, usually to a tape device, for safekeeping.

backup
   A copy of information or data that is kept in case the original is changed or destroyed.

Backup-While-Open
   (BWO) An RUV Subsystem Backup feature that backs up online CICS data sets while leaving
the online data available (or open) to users.

backward file recovery
   The reconstruction of an earlier version of a file by using a newer version and data that have
been recorded in a journal (before-image journal records). Contrast with forward file recovery.

backward recovery
   See backward file recovery.

base cluster
   In systems with VSAM, a key-sequenced or entry-sequenced file over which one or more
alternate indexes are built. See also cluster.

batch
   A group of data processing jobs or records that are stacked together for processing in a
non-interactive environment.
BCSS
BMC Consolidated Subsystem. The BCSS manages I/O to the registration data sets (collectively called the REGISET), manages APF-authorized functions, and performs processing for intercepted open, close, attach, and link requests.

before-image
A journal record of a file transaction (add, delete, or update) before the transaction occurs. Contrast with after-image.

BMCP
BMC Primary Subsystem. The BMCP establishes supervisory services for the BCSS and many BMC Software products. It allows interception of open, close, attach, and link requests in the system.

C
CA
Control Area. The unit of space allocation for a VSAM data set.

CI
Control Interval. A unit of data that is transferred between auxiliary storage and virtual storage, independent of the physical record size.

CDS

command area
The area on the portion of the window which is composed of two elements: a command field prompt and a command entry field.

command entry field
The entry field on the portion of the window where you type commands. The entry field is preceded by a command field prompt.

Common User Access
A set of basic online interface principles, techniques, and components for software applications developed to run on non programmable terminals in IBM’s SAA operating environments.

cluster
In systems with VSAM, a named structure consisting of a group of related components. For example, a data component with its index component. See also alternate index, base cluster.

CSA
Common service area that contains data that is addressable by all address spaces.
CUA
   See Common User Access.

D

data integrity
   Safety of data from unintentional damage or an out-of-synchronization condition.

DBMS
   Database management system.

DTB
   See dynamic transaction backout.

DUPLEX
   The Recovery Utility for VSAM duplicate repository.

duplicate key
   The presence of the same value in a key field or composite key field in more than one record in a data set.

dynamic transaction backout
   The cancellation of uncommitted changes made to a protected resource in the event of a task abending.

E

ECSA
   Extended common service area.

ESDS
   Entry-sequenced data set. In systems with VSAM, a data set whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

F

forward file recovery
   The reconstruction of a file by updating an earlier version with data recorded in a journal (after-image journal records). Contrast with backward file recovery.

forward recovery
   See forward file recovery.
GDG
Generation data group. A collection of data sets with the same base name, such as PAYROLL, that are kept in chronological order. Each data set is called a generation data set.

generation data set
One generation of a generation data group.

IDCAMS
An IBM acronym for the Integrated Catalog Access Method Services facility.

JES
Job entry subsystem. A z/OS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system.

job name
The name assigned to a JOB statement; it identifies the job to the system.

in-flight
Refers to a task that was interrupted before it was completed when a CICS failure or an immediate shutdown occurred. An in-flight task causes records to be written to the system log, but no sync point records are written for the current logical unit of work.

journal
A chronological record of changes made in a data set; the record can be used to reconstruct a previous version of the data set. In CICS, a set of one or more data sets to which records are written sequentially during a CICS run.

journal work data set
A data set created by the Recovery utility for the storage of journal records sorted by file ID, key, and date-time sequence.

journal record
Individual record of transaction changes to a file. Journal records can be used for forward or backward file recovery.

journaling
The process of recording information sequentially in a database.
key sequence
In VSAM, the collating sequence of data records as determined by the value of the key field in each of the data records. The key entry may be the same as, or different from, the entry sequence of the records.

key field
In VSAM, this is a field in each record of a data set that is used as the key for the record. The key identifies the record and establishes the order for the record in respect to other records.

KSDS
Key-sequenced data set. A VSAM file or data set whose records are loaded in key sequence and controlled by an index.

L
logging
The replacement function for batch journaling.

LPA
Link pack area. An area of main storage containing re-enterable routines from system libraries.

LUW
Logical unit of work. The processing that a program performs between synchronization points. See also sync point.

M
master data set
A data set that is used as an authority in a given job and that is relatively permanent, even though its contents may change. For example, a data set, such as PAYROLL, would be the target of a Recovery utility Backup function.

P
PDS
Partitioned data set.

path
In VSAM, a named, logical entity providing access to the records of a base cluster either directly or through an alternate index.

PLT
Program list table.

pop-up
A bordered area of the screen that supplements the dialog that is occurring in the primary window.
**program name**
The name of the program currently executing as identified to system. It is specified in the execute command used to call the program.

**pull-down**
An extension of the action bar that displays a list of available choices for a selected choice in the action bar.

**quiesce**
To end a process by allowing operations to complete in a normal way.

**R**

**RACF**
Resource access control facility.

**RBA**
Relative byte address.

**RCM**
Three-character product code for RECOVERY PLUS for CICS/VSAM.

**read integrity**
Data access which ensures that the most current copy of the record is being read.

**recovery**
The reconstruction of a data base; for example by means of backup files and after-images.

**Recovery Utility for VSAM**
The replacement product for the VSAM Recovery Services product.

**REGISET**
The Recovery Utility for VSAM repository.

**registration**
The recording of information regarding data sets, associated journal records, and backup and recovery information in the Recovery Automation feature’s control data set (CDS). The Recovery Automation feature uses the data to coordinate recoveries.

**repository**
The vehicle for collecting the primary registration data sets.

**rollback**
A programmed return to a prior checkpoint. A rollback cancels the changes to all recoverable resources during the current logical unit of work.
RPCV
   The acronym for RECOVERY PLUS for CICS/VSAM.

RUV
   The common acronym for Recovery Utility for VSAM.

S

SAA
   See Systems Application Architecture.

sequential data set
   A data set that has records organized by successive physical positions. It is typically used for sequential input or output, such as for journal, ledger, and reporting functions.

SHAREOPTION
   An attribute of a VSAM data set which controls the concurrent sharing of its data by different programs running in the same CPU or different CPUs. The types of sharing can be at the read and/or update levels.

SMS
   Storage Management Subsystem. A component of z/OS that automates and centralizes the management of storage.

Systems Application Architecture
   Systems Application Architecture (SAA) is a collection of selected software interfaces, conventions, and protocols published by IBM. It is IBM’s design for creating applications that work in multiple environments. There are four elements to this architecture: Common User Access, Common Programming Interface, Common Communication Support, and Common Applications.

sync point
   An intermediate or end point during processing of a transaction at which an update or modification to one or more of the transaction’s protected resources is logically complete and error free. Synonymous with synchronization point.

synchronization point
   Synonym for sync point.

SYSLOG
   System log. A data set or file for job-related information, operational data, descriptions of unusual occurrences, commands, and messages to or from the operator.

T

task
   A basic unit of work. See also logical unit of work.
VSAM
Virtual Storage Access Method. An access method for direct or sequential processing of fixed and variable length records on direct access devices.
Index

Symbols

$Q04REPO job 565, 566, 567
* comment indicator 82
* masking character 107
<+1> keyword 133
<+1> variable 176, 178, 715
<variable> action variables 179
<variable name> variables 174
? masking character 107

Numerics

32 KB VSAM records 54

A

ACCUM keyword 254
ACCUM processing, reader task settings 54
accumulation
  examples 256
  management overview 66, 68
  merging and grouping data 248
  MUST_COMPLETE keyword 110
  overview 248
  separating data 248
action bar choices
  on Ruleset List panel 209
  standard 117
action variables 179
Activate action bar choice 210
ACTIVATE RULE_SET command 198, 200
activating the EXTERNAL_VENDOR 191
activating the RUV rule set 549
ACTIVE status 69
active status
  setting with ISPF interface 117
  setting with STATUS keyword 112
ADD EXTERNAL_VENDOR command 187
ADD INTERNAL_READER command 184, 186
ADD JOB JCL command 182, 183
ADD JOB_SET command 161, 164
ADD JOURNAL_MODEL command 166, 167, 169, 170
ADD LOG_OF_LOGS command 240, 242
ADD LOG_STREAM command 236, 238
ADD OUTPUT_MODEL command 615
ADD RULE_SET command 192, 193
ADD USER_VARIABLES command 171, 172
ADD VSAM_FILE command 146
ADD VSAM_GROUP_DEFINITION command 601, 603, 627, 629
ADD VSAM_NAME_MODEL command 305
ADD VSAM_SET command 159, 160, 613
after-image journal records 160, 255, 614
AIX keyword 442, 445, 460, 476
allocation parameters for logs and journals 168, 169
alternate indexes rebuilt after restore 460
APPLID keyword
  on ARCHIVE JOURNAL_IN command 228
  on REPORT command 407
APPLID/FILEID
  automatic registration 239
  reports 407
Applid/Field List panel 378
APPLID_FILEID keyword 147, 149
ARCHIVE ACCUM command 250
ARCHIVE ACCUM VSAM_GROUP command 611
ARCHIVE ARCHIVE_IN command 260, 261
archive display of job information in ISPF 389
Archive DSN List panel 387
ARCHIVE JOURNAL_IN command 227, 229
ARCHIVE LOG_STREAM_IN command 230, 233
ARCHIVE processing, reader task settings 54
ARCHIVE RPCV_IN command 272, 273
ARCHIVE UNIT_OF_WORK_IN command 261
ARCHIVE_FILE keyword
  on DELETE ARCHIVE_FILE command 268
  on REPORT command 405
  on UPDATE ARCHIVE_FILE command 265
ARCHIVE_IN keyword
  on ARCHIVE ACCUM command 251
  on ARCHIVE ARCHIVE_IN command 260
  on PRINT ARCHIVE_IN command 274
  on RECOVER command 477, 485, 491
  on REGISTER ARCHIVE_IN command 262
ARCHIVE_OUT data set default DCB attributes 80
ARCHIVE_OUT keyword
  on ARCHIVE ACCUM command 256
  on ARCHIVE ARCHIVE_IN command 261
  on ARCHIVE JOURNAL_IN command 228
  on ARCHIVE LOG_STREAM_IN command 232
archives
accumulating 248
ACTIVE status 69
automatic management 52
CICS journals 227
CICS-only requirements 217
collecting and managing 66
copying 69
copying and moving 260
deleting 268
management overview 65
managing 247
merging and grouping data 248
moving 69
printing data 274
PROGRES status 69
registering 262
reports 367, 405
RPCV 272
separating data 248
status 112
Transaction Server log stream 230
updating 264
ASSOCIATED_ADDRESS_SPACE_TASKS 126
asterisk, use for comments 82, 107
AUDIT function 433
auditing 432
automatic batch backout
generated JCL 164
identifying programs to use 59
internal readers 164, 184
introduction 51
planning 60
skeleton JCL 163, 182
BACKOUT_FILE keyword (continued)
on REPORT BACKUP_FILE command 270, 369
on REPORT command 408
on UPDATE BACKUP_FILE command 365
BACKUP_IN keyword
on RECOVER FORWARD command 478, 485
on REGISTER BACKUP_IN command 362
on RESTORE REPOSITORY command 574
on RESTORE VSAM_FILE command 459, 461, 480
BACKUP_METHOD keyword 126, 371, 409
BACKUP_OUT data set
coding DD statement 291
default DCB attributes 80
BACKUP_OUT keyword 441
on BACKUP REPOSITORY command 574
on BACKUP_VSAM command 293
BACKUP_REGISTRATION keyword 188
<BACKUP_TIME> variable 174, 713
backups
for disaster recovery 527
RBA0 312
Backup-While-Open Feature
changing ownership of coordinator 553
controlling 221
implementation planning 75
management overview 76
restoring fuzzy backups 451
batch interface
backout recovery 490
batch recovery, performing 489
commands 86
forward recovery 473, 474, 483, 484, 489
help for problems 84
keyword values 105
keywords 91
reports 401
using 79
Batch Journaling Facility
disable in CICS-only environments 221
implementation planning 57
implementing 154
introduction 50
Batch Journaling Facility, planning for 57
batch logs, <+1> variable 178
batch programs
automatic batch backout 51, 60
determining recovery actions 59
enabling recovery 59
excluding from journaling 162
excluding from logging 162
excluding from recovery 59
job failures 163
journaling 50
journaling failures 163
setting up rules 161
use of rule sets 199
Backout-While-Open Feature
changing ownership of coordinator 553
controlling 221
implementation planning 75
management overview 76
restoring fuzzy backups 451
batch interface
backout recovery 490
batch recovery, performing 489
commands 86
forward recovery 473, 474, 483, 484, 489
help for problems 84
keyword values 105
keywords 91
reports 401
using 79
Batch Journaling Facility
disabling in CICS-only environments 221
implementation planning 57
implementing 154
introduction 50
Batch Journaling Facility, planning for 57
batch logs, <+1> variable 178
batch programs
automatic batch backout 51, 60
determining recovery actions 59
enabling recovery 59
excluding from journaling 162
excluding from logging 162
excluding from recovery 59
job failures 163
journaling 50
journaling failures 163
setting up rules 161
use of rule sets 199

Index 733

A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z

BCSS
changing BWO ownership 553
commands data set 544
console commands 544
displaying BCSS status 549
displaying diagnostic information 553
displaying help 550
displaying RUV status 552, 556
displaying validation information 554
enabling RUV activity 550, 556
hardware reserve for REGISET 560
multiple 536
overview 536
procedure names 537
REGISET usage 559
starting 548
stopping 550
stopping RUV activity immediately 552, 556
stopping RUV activity normally 551, 556
subsystem IDs 537
using 543
validation of CPU authorization passwords 538
before-image journal records 160, 255
<BEGIN_LOOP> variable 179, 715
BLKSIZE attribute of output data sets 80
BMC Consolidated Subsystem parameters 545
BMC primary subsystem parameters 540
BMC Software, contacting 2
BMC subsystem parameters 537
BMCP
  console commands 539
displaying status 541
overview 535
procedure names 537
starting 541
stopping 542
subsystem IDs 537
using 539
buffers, flushing 168
BUFND keyword 478, 491
BUFNI keyword 478, 491
BWO, controlling 221

C
Change Status action bar choice 117
check of syntax with SYNTAX_CHECK keyword 137
CICS
archiving journal extents 227
  journal data archival 50
mapping APPLID to DSN 228
requirements 220
CICS TS environment, coding file definitions 218
CICS-only environments
  archive requirements 217
  archiving journals 227

CICS-only environments (continued)
  automatic creation of APPLID/FILEID records 239
  backout recovery procedure 226
  backup requirements 221
  disabling the Batch Journaling Feature 221
  forward recovery procedure 224, 225
  management overview 75
CLASS keyword 185
clock date/time conversion 121
combined archives 248
command
  BACKUP SEQ_GROUP 629
  BACKUP VSAM_GROUP 603
command language elements 82
Command Mode action bar choice 118
command summary 86
command usage 86
commands
  ACTIVATE RULE_SET 198, 200
  ADD EXTERNAL_VENDOR 187
  ADD INTERNAL_READER 184, 186
  ADD JOB_JCL 182, 183
  ADD JOB_SET 161, 164
  ADD JOURNAL_MODEL 166, 167, 169, 170
  ADD LOG_OF_LOGS 240, 242
  ADD LOG_STREAM 236, 238
  ADD OUTPUT_MODEL 615
  ADD RULE_SET 192, 193
  ADD USER_VARIABLES 171, 172
  ADD VSAM_FILE 146
  ADD VSAM_GROUP_DEFINITION 601, 603, 627, 629
  ADD VSAM_NAME_MODEL 305
  ADD VSAM_SET 159, 160, 613
  ARCHIVE ACCUM 250
  ARCHIVE ACCUM VSAM_GROUP 611
  ARCHIVE ARCHIVE_IN 260, 261
  ARCHIVE JOURNAL_IN 227, 229
  ARCHIVE LOG_STREAM_IN 230, 233
  ARCHIVE RPCV_IN 272, 273
  BACKUP REPOSITORY 573, 576
  BACKUP VSAM_GROUP 310, 603, 629
  BACKUP VSAM_IN 310
  DEBUG 553
  DELETE ARCHIVE_FILE 268, 269
  DELETE BACKUP_FILE 367
  DELETE EXTERNAL_VENDOR 191
  DELETE EXTERNAL_VENDOR keyword 212
  DELETE INTERNAL_READER 212
  DELETE JOB_JCL 212
  DELETE JOB_SET 212
  DELETE JOURNAL_MODEL 212
  DELETE LOG_OF_LOGS 237, 240, 241, 243
  DELETE LOG_STREAM 237, 238
  DELETE OUTPUT_MODEL 617
  DELETE OUTPUT_MODEL keyword 212
  DELETE RULE_SET 212
  DELETE USER_VARIABLES 212
commands (continued)
    DELETE VSAM_FILE 150
    DELETE VSAM_GROUP 619, 634
    DELETE VSAM_GROUP_DEFINITION 618, 633
    DELETE VSAM_NAME_MODEL 212, 305
    DELETE VSAM_SET 212
    DISABLE 552, 556
    ENABLE 550, 556
    ENTER_IDCAMS 514
    EXIT_IDCAMS 514
    HELP 550
    PRINT ARCHIVE_IN 274, 276
    PRINT LOG_STREAM_IN 234, 235
    PRINT REPOSITORY 577
    PURGE REPOSITORY 588, 590
    QUIESCE 551, 556
    RECOVER BACKOUT 489, 490
    RECOVER BACKOUT VSAM_GROUP 610
    RECOVER FORWARD 473, 474, 483, 484, 489
    RECOVER FORWARD VSAM_GROUP 607, 609
    RECOVER REPOSITORY FORWARD 570, 571
    REGISTER ARCHIVE_IN 262, 263
    REGISTER BACKUP_FILE 355
    REGISTER BACKUP_IN 362
    REGISTER LOG_OF_LOGS 239, 243
    REGISTER RESTORE_JOB_JCL 357
    REGISTER RESTORE_METHOD 359
    REINIT 548
    REPORT 401
    REPORT ARCHIVE_FILE 270
    REPORT AUDIT 433, 434
    REPORT BACKUP_FILE 271, 369
    REPORT BACKUP_METHOD 371
    REPORT DEFAULT 142
    REPORT OUTPUT_MODEL 617
    REPORT SYSTEM_VARIABLES 178
    REPORT VSAM_GROUP 621, 636
    REPORT VSAM_GROUP_DEFINITION 620, 635
    REPORT VSAM_NAME_MODEL 305
    RESET DEFAULT 142
    RESTORE REPOSITORY 574, 576
    RESTORE SEQ_FILE 462
    RESTORE SEQ_GROUP 631
    RESTORE VSAM_FILE 462
    RESTORE VSAM_GROUP 605, 631
    RUV DEBUG 553
    RUV DISABLE 552, 556
    RUV ENABLE 550, 556
    RUV HELP 550
    RUV QUIESCE 551, 556
    RUV STATUS 552, 556
    RUV TAKEOVER BWOCOORD 553
    SET 125, 144
    SET DEFAULT 124, 143
    SHUTDOWN 542, 550
    STATUS 541, 549, 552, 556
    TAKEOVER BWOCOORD 553

commands (continued)
    TEST DELAY 515
    TRY 201
    TSO RUVISPF 113
    UPDATE ARCHIVE_FILE 264, 266
    UPDATE BACKUP_FILE 363
    UPDATE EXTERNAL_VENDOR 188
    UPDATE INTERNAL_READER 186
    UPDATE LOG_OF_LOGS 240, 243
    UPDATE LOG_STREAM 236, 238
    UPDATE OUTPUT_MODEL 617
    UPDATE SEQ_GROUP_DEFINITION 633
    UPDATE VSAM_FILE 148
    UPDATE VSAM_GROUP 619, 634
    UPDATE VSAM_GROUP_DEFINITION 618
    UPDATE VSAM_NAME_MODEL 305
    VALIDATE REPOSITORY 591
    VALIDATE SUBSYSTEM 554

COMMENT keyword 107
comments, use of asterisk 82
COMPLETE_COPY keyword 228, 232, 272
compressed ESDSs 284
compressed VSAM files 53
condition code control 137
considerations for using RUV 53
calendar command summary 549
consolidated subsystem, activating the RUV rule set 549
CONSOLIDATED keyword 254
consolidated subsystem parameters, BMC 545
conversion
    document 33
    syntax statements 35
typographical 33
conversion
    Julian date 120
timestamps 121
conversion of RPCV to RUV 647
COPY BACKUP command 346
copy of archives 260, 261
COUNT keyword 274, 577
creating a user-defined date format 130
CURRENT_BACKUP 449, 461
customer support 3
cycles keyword 129

D

D SYMBOLS command 178
data set names 80
data set status 112
date
    <RUVDATE> variable 175, 713
calendar of Julian 120
conversion of Julian 120
format preference 122, 129, 130
Index 735

Date Conversion action bar choice 118, 120
Date Conversion popup 120
DATE_FORMAT keyword 122, 129, 130
daylight savings time considerations 222
DAYS keyword
   on REGISTER LOG_OF_LOGS command 242
   on SET-type command 131
DCB attributes of output data sets 80
DDNAME keyword 169, 185
ddnames 80
DEBUG command 553
DEFAULT keyword
   on REPORT command 410
   on SET DEFAULT command 124
default options 125
   report 410
   reporting 142
   resetting 142
   setting 124
defining groups 601, 627
defining sequential groups 624
defining VSAM groups 596
DELAY keyword 515
Delete action bar choice 117, 209
DELETE ARCHIVE_FILE command 268, 269
DELETE BACKUP_FILE command 367
DELETE command, ISPF interface 117
DELETE EXTERNAL_VENDOR command 191
DELETE INTERNAL_READER command 212
DELETE JOB_JCL command 212
DELETE JOB_SET command 212
DELETE JOURNAL_MODEL command 212
DELETE keyword
   on ARCHIVE LOG_STREAM_IN command 231
   on PRINT LOG_STREAM_IN command 234
   on REGISTER LOG_OF_LOGS command 241
DELETE LOG_OF_LOGS command 237, 240, 241, 243
DELETE LOG_STREAM command 237, 238
DELETE OUTPUT_MODEL command 212, 617
DELETE RULE_SET command 212
DELETE USER_VARIABLES command 212
DELETE VSAM_FILE command 150
DELETE VSAM_GROUP command 619, 634
DELETE VSAM_GROUP_DEFINITION command 618, 633
DELETE VSAM_NAME_MODEL command 212, 305
DELETE VSAM_SET command 212
destination of data 110
detail-level reports 135, 401
diagnostic procedures 639
DISABLE command 552, 556
disaster recovery 523
   evaluating 531
   overview 524
   performing 529
   preparing for 526
Display Filter popup window 119
document conventions 33
documentation of commands and data sets 107
documentation, online 32
DSN Detail - Applid/Fileid List panel 380
DSN Detail - Backup List panel 384, 385
DSN Detail - Comments panel 382
DSN Detail - Path List panel 383
DSN Detail - User Recovery Exit List panel 381
DSN keyword 201
DSORG attribute of output data sets 80
duplex registration data set
   overview 559
   recovery 567
   recovery of primary 565
   usage 560

E
Edit action bar choice 209
electronic documentation 32
ENABLE command 550, 556
END_DATA keyword 171, 183, 330
<<END_LOOP> variable 179, 715
enqueue, file 515
ENTER_IDCAMS command 514
ESDS, compressed 53, 284
ESOTERIC keyword 131
EXCLUDE
   from journaling 162
   from logging 162
   from recovery 160
Exit action bar choice 117
exit, user 517
EXIT_IDCAMS command 514
EXTERNAL_VENDOR activating 191
deleting 212
EXTERNAL_VENDOR keyword 193
EXTERNAL_VENDOR report 411

F
features of RUV 47
File action bar choice 117
file definitions, CICS
   requirements 217
   specifying parameters 218
file definitions for CICS TS environment, coding 218
Filter action bar choice 118
FILTER_BY keyword 108
Flip View action bar choice 118
forms for information gathering 661
forward recovery
   actions 467
forward recovery (continued)
  automatic restore of VSAM file 448
  automation 465
  batch interface 473, 474, 483, 484, 489
  enabling VSAM_RULE keyword 160, 614
  IDCAMS delete/define 514
  introduction 42, 51
  ISPF interface 495
  manual restore 466
  overview 465
  performing 473
  procedure in CICS-only environments 224, 225
  RLS protection 135
  with ISPF panels 376
full-level reports 135, 401
fuzzy backups 451

G

GAP keyword 131, 232
GMT timestamps
  <RUV_STEPLIB> variable 176, 714
  <RUVTIME> variable 176, 714
  <RUVTIME_HEX> variable 176, 714
  conversion 121
  RUV usage 111
groups
  concepts 596, 624
  definition 597, 624
  disaster recovery 526
  information through ISPF display 393
  maintenance 599, 625
  using 597, 603, 624, 629

H

hardware reserve 560
HELP command 550
help, batch interface 84
HOSTCOPYMODE keyword 109

I

I/O overlapping 54
IAM, disk management of VSAM files support 48
IBM-defined variables 178
IDCAMS delete/define
  during recovery 467, 514
  during restore 451
  <IMBED> variable 179, 716
inactive status
  setting with ISPF interface 117
  setting with STATUS keyword 112
INCREMENTAL_BACKUP_SUPPORT, keyword 132
Inquiry Mode action bar choice 117
insertion strategy, controlling 134
Instant Snapshot copies
  creating 302
  esoteric devices 131
  introduction 50
internal readers
  creating 184
  creating user variables 171
  deleting 212
  processing 199
  reports 412
  using variables 173
INTERNAL_READER keyword 212
  on ADD INTERNAL_READER command 184
  on ADD JOB_SET command 164
  on REPORT command 412
INVENTORY_ONLY keyword 251, 293, 296, 342, 348, 435,
  441, 444, 482, 486, 604, 605, 607, 609, 610, 630, 631
ISPF interface
  action bar choices 117
  archive detail 389
  group and output model information 393
  limiting table displays 119
  setting preferences 122
  starting 113
  subsystem information 555
  using 113
  viewing mode preference 122
  viewing reports 375
  VSAM file recovery 495
  working with rule sets 205

J

JCL
  automatic batch backout 182
  generation preference 122
  JOB_JCL keyword 163
  RUVZSM0 utility 79
  tailoring preference 122
JCT
  requirements 217
  table conversion 647
JOB keyword 201
job sets
  creating 161
  deleting 212
  information gathering forms 663
  overview 63
  processing 199
  reports 414
  use in rule set 157
JOB_ABEND keyword 163
JOB_JCL keyword 182, 212
  on ADD JOB_SET command 163
  on REPORT command 413
<JOB_NUMBER> variable 175, 713
JOB_RULE keyword 161
JOB_SET keyword 212
  on ADD RULE_SET command 192
  on REPORT command 414
job-level options 125
job-level rule sets 63, 154
<JOBNAME> variable 174, 713
jobs, monitoring 400
JOURNAL keyword 162
JOURNAL_MANAGER PLUS product 227
journal models
  creating 166
  creating user variables 171
  deleting 212
  information gathering forms 664
  planning 60
  processing 199
  reports 415
  use on JOURNAL keyword 162
  use on LOG keyword 162
  using variables 173
JOURNAL_ABEND keyword 163
JOURNAL_IN keyword 228
JOURNAL_MODEL keyword 212
  on ADD JOURNAL_MODEL command 166, 167
  on REPORT command 415
JOURNAL_REGISTRATION keyword 189
journals
  <RUV_JOURNAL> variable 175, 714
  <RUV_JOURNAL_DEACTIVATE> variable 175, 265, 714
  archive introduction 50
  archiving 227
  contents and disposition 60
  data set name 61
  default DCB attributes 80
  failure action 163
  planning 60
  RPCV 272
  size 61
Julian date conversion 120

K

keyword
  RETAIN_BACKUP 442, 445
  VSAM_NAME_MODEL 442, 446
keyword values
  summary 105
  usage 105
  use data set name instead of ddname 80
keywords
  <+1> 133
  ACCUM 254
  AIX 442, 445, 460, 476
keywords (continued)
  APPLID 228, 407
  APPLID_FILEID 147, 149
  ARCHIVE_FILE 265, 268, 405
  ARCHIVE_IN 251, 260, 274, 477, 485, 491
  ARCHIVE_OUT 228, 232, 256, 261
  ASSOCIATED_ADDRESS_SPACE_TASKS 126
  BACKOUT_OUT 569
  BACKUP_CHANGED_ONLY 126
  BACKUP_FILE 270, 355, 365, 367, 369, 408
  BACKUP_IN 362, 459, 461, 478, 480, 485, 574
  BACKUP_METHOD 126, 371, 409
  BACKUP_OUT 293, 441, 574
  BACKUP_REGISTRATION 188
  BUFND 478, 491
  BUFNI 478, 491
  CLASS 185
  COMMENT 107
  COMPLETE_COPY 228, 232, 272
  CONSOLIDATED 254
  COUNT 274, 577
  CURRENT_BACKUP 449, 461
  CYCLES 129
  DATE_FORMAT 122, 129, 130
  DAYS 131, 242
  DNAME 169, 185
  DEFAULT 124, 410
  DELAY 515
  DELETE 231, 234, 241
  DSN 201
  END_DATA 171, 183, 330
  ESOTERIC 131
  EXTERNAL_VENDOR 193, 212
  FILTER_BY 108
  GAP 131, 232
  HOSTCOPYMODE 109
  INCREMENTAL_BACKUP_SUPPORT 132
  INTERNAL_READER 164, 184, 212, 412
  INVENTORY_ONLY 251, 293, 296, 342, 348, 435, 441, 444, 482, 486, 604, 605, 607, 609, 610, 630, 631
  JOB 201
  JOB_ABEND 163
  JOB_JCL 163, 182, 413
  JOB_RULE 161
  JOB_SET 192, 212, 414
  JOURNAL 162
  JOURNAL_ABEND 163
  JOURNAL_IN 228
  JOURNAL_MODEL 166, 167, 415
  JOURNAL_REGISTRATION 189
  LANGUAGE 122, 132
  LAST_IMAGE 255, 612
  LINE_COUNT 133
  LOCATION 110
  LOG 162
  LOG_OF_LOGS 241, 416
  LOG_STREAM 237, 418
keywords (continued)
LOG_STREAM_IN 230, 234
LSR_POOL 479, 492
MAX_CC 133
MEMBER 185
MERGE 575
MUST_COMPLETE 110
NEW_NAME 449, 459, 480, 481
OUTPUT_MODEL 212, 419, 445
PAGE_WIDTH 133
PASSWORD_EXPIRATION_CC 133
PDS_OUT 185
PREFIX 292, 296
PROGRAM 201
PROTECTION_LEVEL 168
PURGE_REPOSITORY VSAM_GROUP 621, 636
PURGE_OUT 589
QSAM 168
READER_TASKS 54, 134, 286
RECOVERY_MODE 134, 480, 492
REGISTRATION_TIME 265, 365
REPLACE 306
REPORT 135
RETRIEVE_IMAGE 255
RLS_RECOVER_PROTECTION 135
RPCV_IN 272
RULE_SET 198, 212, 420
SCRATCH 252, 269, 368, 589
SELECTION_EXIT 147, 149, 480, 493, 517
SEQ_FILE 355, 459
SIMULATE 589
SMF_IN 569
SMF_TYPE 568
START_TIME 111, 234, 238, 242, 275, 355, 357, 360, 435, 460, 481, 486, 493, 575, 606, 632
STATUS 112, 188
STOP_TIME 110, 275, 435, 460, 481, 487, 493, 575, 606, 632
SUBSYSTEM 136
SUPPRESS_RECOVERY_ERROR 482
SYNTAX_CHECK 137
SYSTEM_VARIABLES 422
TERM_CC 137
TERMID 436
TIME_FORMAT 122, 138
TRACE_ON 199, 201, 549
TRANID 436
UNSELECTED_VSAM_FILES 254
UPPER_CASE_TRANSLATION 138
USE_AUTOJOURNAL 139
USER_VARIABLES 192, 212, 411, 424
VSAM_FILE 147, 149, 254, 355, 357, 359, 425, 458, 482, 488, 494, 589
VSAM_FILE_MASK 306
VSAM_GROUP 254, 427, 444, 589
VSAM_GROUP_DEFINITION 212, 428
VSAM_IN 292, 296, 440

keywords (continued)
VSAM_NAME_MODEL 212, 305, 429
VSAM_RULE 159, 614
VSAM_SET 192, 212, 430
WORK_DISK 141
WORK_TAPE 141
XBMID 141, 431

L

LANGUAGE keyword 122, 132
language preference 122, 132
last image accumulation of VSAM file archives 48
LAST_IMAGE keyword 255, 612
libraries, RUV 36
LINE_COUNT keyword 133
load library 36
local recovery
LOCATION keyword 110
management overview 73
LOCATION keyword 110
LOG keyword 162
log models
creating 166
creating user variables 171
planning 60
processing 199
using variables 173
log of logs
managing records in repository 239
reports 391, 392, 416
Log of Logs Detail panel 392
Log of Logs Selection panel 391
log streams
archiving 230
GAP keyword 131, 232
missing data 131, 232
printing 234
reports 418
LOG_OF_LOGS keyword
on DELETE LOG_OF_LOGS command 241
on REPORT command 416
LOG_STREAM keyword
on ADD, UPDATE, and DELETE LOG_STREAM commands 237
on REPORT command 418
LOG_STREAM_IN keyword
on ARCHIVE command 230
on PRINT command 234
logs
<RUV_LOG> variable 176, 714
contents and disposition 60
data set name 61
default DCB attributes 80
failure actions 163
size 61
<LOOP_TIME> variable 175, 713
<LOOP_TIME_HEX> variable 175, 713
LRECL attribute of output data sets 80
LSR_POOL keyword 479, 492

M
maintaining groups 615, 633
masking characters
usage in commands 107
usage on Display Filter panel 119
MAX_CC keyword 133
MEMBER keyword 185
MERGE keyword 575
merge of archives 248
Message Lookup panel 398
Message Text panel 399
messages, displaying with ISPF interface 398
move of archives 260
Multiple Primary feature 559
multiple processes 54
multi-tasked reader process 134, 286
multitasking 54
MUST_COMPLETE keyword 110

N
naming temporary disk allocations 141
naming temporary tape allocations 141
New action bar choice 209, 211
NEW_NAME 481
NEW_NAME keyword 449, 459, 481
<NODEn> variable 175, 713
non-RUV backups, restore 450

O
online documentation 32
online help 32
OPT table conversion 647
OUTPUT 445
output data set default DCB attributes 80
output model information through ISPF display 393
OUTPUT_MODEL keyword 212, 419, 445

P
page length 133
PAGE_WIDTH keyword 133
panels
  Applid/Fileid List 378
  Archive DSN List 387, 389
  Date Conversion 120
  Display Filter 119
panels (continued)
  DSN Detail - Applid/Fileid List 380
  DSN Detail - Backup List 384, 385
  DSN Detail - Comments 382
  DSN Detail - Path List 383
  DSN Detail - User Recovery Exit List 381
  Log of Logs Detail 392
  Log of Logs Selection 391
  Message Lookup 398
  Message Text 399
  PTF List 641
  Ruleset List 206
  RUV Job Monitoring 400
  RUV Options 115
  RUV Ruleset/Group/Model Templates 211
  Subsystem Detail 395, 555
  Subsystem List 113
  TOD Clock Date/Time Conversion 121
  User Preference 122
  VSAM DSN Detail - Base Cluster 379
  VSAM DSN List 376
<PARAMETER> variable 175, 713
parameters, consolidated subsystem
  CAPS 546
  CASPROC 547
  DEBUG 546
  DSPSIZE 547
  FORCESYN 547
  MIXED 546
  NODEBUG 546
  NODUPELEX 548
  NOPROMPT 546
  NOREFRESH 546
  NOREGISSET 548
  PRIVATE 547
  PROMPT 546
  PUBLIC 547
  REFRESH 545
  SKIPSYN 546
  SUBSYSID 547
parameters, general 537
parameters, primary subsystem 540
  CAPS 540
  DEBUG 540
  MIXED 540
  NODEBUG 540
  PROMPT 540
  REFRESH 540
  SMFRECID 540
  SUBSYSID 541
  SVCNO 541
PASSWORD_EXPIRATION_CC keyword 133
PDS_OUT data set default DCB attributes 80
PDS_OUT keyword 185
performance, PROTECTION_LEVEL keyword 168
<PGMNAME> variable 175, 713
planning for the Batch Journaling Facility 57
<+1> keyword 133
<+1> variable 176, 177, 715
point-in-time recovery 466
point-in-time restore 449
preferences, setting 122
PREFIX keyword 292, 296
primary registration data set
  overview 559
  recovery 564
  usage 559
primary subsystem parameters 540
PRINT ARCHIVE_IN command 274, 276
PRINT LOG_STREAM_IN command 234, 235
PRINT REPOSITORY command 577
printing archive data to an external vendor 189
problem solving
  COMPLETE_COPY keyword 228, 232
  procedures 638
<PROCNAME> variable 175, 713
product support 3
PROGRAM keyword 201
PROGRES status 69
PROTECTION_LEVEL keyword 168
PTF List panel 641
PTF report 641
purge of repository records 579
purge processing
  CYCLES keyword 129
  DAYS keyword 131
PURGE REPOSITORY command 588, 590
PURGE REPOSITORY VSAM_GROUP keyword in groups 621, 636
PURGE_OUT data set
  default DCB attributes 80
  using 588
PURGE_OUT keyword 589

Q
QSAM keyword 168
question mark masking character 107
QUIESCE command 551, 556

R
RBA0 backups 312
READER_TASKS keyword 134, 286
RECFM attribute of output data sets 80
record counts in reports 177
RECOVER BACKOUT command 489, 490
RECOVER BACKOUT VSAM_GROUP command 610
RECOVER FORWARD command 473, 474, 483, 484, 489
RECOVER FORWARD VSAM_GROUP command 607, 609
RECOVER processing, reader task settings 54
RECOVER REPOSITORY FORWARD command 570, 571
recovering an SMF data set archive 568
recovery
  determining recovery actions 59
  disaster 523
  evaluating 531
  introduction 39
  overview 524
  performing 529
  preparing for 526
  process 41
REGISET 564
setting type with VSAM_RULE keyword 159, 614
with ISPF panel 376
RECOVERY_MODE keyword 134, 480, 492
Refresh action bar choice 118
REGISET
  analysis 572
  backups 572
duplex registration data set 559, 560
hardware reserve 560
initialization 561
Multiple Primary feature 559
overview 559
primary registration data set 559
records 561
recovering from backup 566
recovery 564
registration data sets 559
serialization process 561
SHAREOPTIONS 560
REGISTER ARCHIVE_IN command 262, 263
REGISTER BACKUP_FILE command 355
REGISTER BACKUP_IN command 356
REGISTER LOG_OF_LOGS command 239, 243
REGISTER RESTORE_JOB_JCL command 357
REGISTER RESTORE_METHOD command 359
REGISTRATION_TIME keyword 265, 365
REINIT command 548
related documentation 32
remote recovery
  LOCATION keyword 110
  management overview 74
REORGANIZE function
  groups 443
  individual files 440
  using 439
REORGANIZE VSAM_GROUP command 444
REORGANIZE VSAM_IN command 440
reorganizing VSAM files
  groups 443
  individual files 440
  overview 437
REPLACE keyword 306
REPORT ARCHIVE_FILE command 270
REPORT AUDIT
  command 433, 434
  function 433
RUV ENABLE command 550, 556
RUV HELP command 550
RUV Job Monitoring panel 400
RUV libraries 36
RUV Options panel 115
RUV QUIESCE command 551, 556
RUV Ruleset/Group/Model Templates panel 211
RUV STATUS command 552, 556
RUV TAKEOVER BWOCOORD command 553
<RUV_CONTROL> variable 175, 713
<RUV_JOB_JCL> variable 175, 713
<RUV_JOURNAL> variable 175, 714
<RUV_JOURNAL_DEACTIVATE> variable 175, 265, 714
<RUV_LOG> variable 176, 714
<RUV_BACKUP> variable 175, 713
<RUVDATE> variable 175, 713
RUV-defined variables
   listing 178
   summary 174
RUVISPF command 113
RUVM transaction 221
RUVSTAT program 189
<RUVTIME> variable 176, 714
<RUVTIME_HEX> variable 176, 714
RUVTRACE data set default DCB attributes 80
RUVZCDA utility 649
RUVZCJC utility 647, 648
RUVZSM0 utility 79
RUVZVML utility 536

S

S BMCP command 541
   sample library 36
Sarbanes-Oxley (SOX) reporting 53, 432
SCRATCH keyword
   on ARCHIVE ACCUM command 252, 269
   on DELETE BACKUP_FILE command 368
   on PURGE REPOSITORY command 589
SELECTION_EXIT keyword 147, 149, 480, 493, 517
   semicolon terminator of commands 82
   separators
   START_TIME and STOP_TIME keywords 111
      use in commands 82
SEQ_FILE keyword 355
   on RESTORE SEQ_FILE command 459
   sequential file backups
      benefits 280
      groups 297
      individual files 294
   sequential groups
      concepts 624
      maintaining 633
      using 629
   serialization process, REGISET 561
   SET command 122, 125, 144
   SET DEFAULT command 124, 143
   SET DEFAULT SMF_TYPE(x,y) 568
   SET SMF_TYPE(x,y) command 568
   sets, overview 62
   sharing control data set 135
   SHCDS 135
   SHUTDOWN command 542, 550
   SIMULATE keyword 589
   SMF, recovering data set archive 568
   SMF_IN 569
   <SMFID> variable 176, 714
   SMS backup restore 451
   SNAPSHOT UPGRADE FEATURE
      instant snapshot function 302
      XBM subsystem 141
   SORT 54
   sort parameters
      increasing memory 472
      setting user preferences 122
   specifying BMC subsystem parameters 537
   specifying parameters, CICS file definitions 218
   sphere recovery selection list 376
   START_TIME keyword 234, 238, 435
      on PRINT ARCHIVE_IN command 275
      on RECOVER command 481, 486, 493
      on REGISTER BACKUP_FILE command 355, 357, 360
      on REGISTER LOG_OF_LOGS command 242
      on RESTORE REPOSITORY command 575
      on RESTORE SEQ_GROUP command 632
      on RESTORE VSAM_FILE command 460
      on RESTORE VSAM_GROUP command 606
      usage 111, 234
   status check utility, RUVZVML 536
   STATUS command 541, 549, 552, 556
   STATUS keyword 112, 188
   <STEPNAME> variable 176, 714
   STOP_TIME keyword 435
      on PRINT ARCHIVE_IN command 275
      on RECOVER command 481, 487, 493
      on RESTORE REPOSITORY command 575
      on RESTORE SEQ_GROUP command 632
      on RESTORE VSAM_FILE command 460
      on RESTORE VSAM_GROUP command 606
      usage 110
   substitution values of variables 171
   substring notation for variables 174

742    Recovery Utility for VSAM User Guide
subsystem
   batch utility selection 136
   connection to ISPF interface 113, 115
   console command summary 123
   entering commands with ISPF interface 118
   overview 535
   procedure names 537
   sharing with BMC Software products 535
   Subsystem Detail panel 395
   subsystem IDs 537
   SUBSYSTEM keyword 136
   Subsystem Detail panel 395, 555
   SUBSYSTEM keyword 136
   Subsystem List popup 113
   subsystem name, overriding 79
   subsystem parameters 537, 545
   subsystems, XBM 141
   subtasking 54
   summary-level reports 135, 401
   support, customer 3
   supported VSAM file types 40, 53
   SUPPRESS_RECOVERY_ERROR keyword 482
   syntax conventions for commands 83
   syntax statement conventions 35
   SYNTAX_CHECK keyword 137
   <$SYNSDX$> variable 176, 714
   SYSPRINT data set default DCB attributes 80
   system variables reports 422
   SYSTEM_VARIABLES keyword 422
   system-level options
      reporting 142
      resetting 142
      setting 124
   system-level rule sets 63, 154

T

TAKEOVER BWOCOORD command 553
   technical support 3
   TERM_CC keyword 137
   TERMDID keyword 436
   terminator for commands 82
   TEST DELAY command 515
   test of rule sets 201
   time format preference 122, 138
   TIME_FORMAT keyword 122, 138
   timestamps
      <$RUV_STEPLIB$> variable 176, 714
      <$RUVTIME$> variable 176, 714
      <$RUVTIME_HEX$> variable 176, 714
      conversion 121
      earliest 111, 234
      generation 466
      latest 110
   TOD action bar choice 118, 121
   TOD Clock Date/Time Conversion popup 121
   Tools action bar choice 118
   trace facilities 639
   TRACE_ON keyword 199, 201, 549
   TRANID keyword 436
   Transaction Server
      archiving log streams 230
      managing log of logs records in repository 239
      printing log streams 234
   TRY command 201
   TSO RUVISPF command 113
   typographical conventions 33

U

UNKNOWN VSAM files 146
UNSELECTED_VSAM_FILES keyword 254
UOW recovery
   introduction 43
   performing 483
   UPDATE_ARCHIVE_FILE command 264, 266
   UPDATE_BACKUP_FILE command 363
   UPDATE_EXTERNAL_VENDOR command 188
   UPDATE_INTERNAL_READER command 186
   UPDATE_LOG_OF_LOGS command 240, 243
   UPDATE_LOG_STREAM command 236, 238
   Update Mode action bar choice 118
   UPDATE_OUTPUT_MODEL command 617
   UPDATE_SEQ_GROUP_DEFINITION command 633
   UPDATE_VSAM_FILE command 148
   UPDATE_VSAM_GROUP command 619, 634
   UPDATE_VSAM_GROUP_DEFINITION command 618
   UPDATE_VSAM_NAME_MODEL command 305
   UPPER_CASE_TRANSLATION keyword 138
   USE_AUTOJOURNAL keyword 139
   user exit 517
   User Preference action bar choice 118, 122
   User Preference panel 122
   user variables 156
      associating with rule sets 192
      creating 171
      defining 171
      deleting 212
      planning 61
      processing 199
      reports 411, 424
      types 61
      using 173
   USER_VARIABLES keyword 212
      on ADD RULE_SET command 192
      on REPORT command 411, 424
      user-defined date format, creating 130
   <$USERID$> variable 176, 714
      using the BACKUP_SEQ_GROUP command in groups 629
      using the BACKUP_VSAM_GROUP command in groups 603

Index  743
V

VALIDATE REPOSITORY command 591
VALIDATE SUBSYSTEM command 554
variables 156
  action 179
  IBM-defined 178
  management overview 62
  names 171
  RUV-defined 174, 178
  substitution value 171
  system reports 422
  user 171
  user variables reports 411, 424
  using 173
  using substring notation 174
View action bar choice 117
View Mode preference 122
VSAM backups
  automatic management 52
  CICS-only requirements 221
  introduction 49
  management overview 70, 71
  output data set 291
  prefixed 292, 296
  procedure 289, 473, 483, 489
  reducing frequency 64
  reports 389, 408
  stacking 70
  status 112
VSAM DSN Detail - Base Cluster panel 379
VSAM DSN List panel 376
VSAM files
  backup procedures 289, 473, 483, 489
  CICS-only support 217
  compressed 53
  enabling recovery 159
  excluding from recovery 58, 160
  IDCAMS delete/define during recovery 514
  identifying for Batch Journaling Facility 57
  managing repository records 146
  recovery 465
  reports 376, 425
  restore overview 448
  selecting types of recovery 58
  status 112
  types not supported by RUV 53
  types supported by RUV 40
  UNKNOWN listing in repository 146
VSAM groups
  concepts 596
  defining 601, 627
  definition 597, 624
  maintaining 615
  maintenance 599, 625
  using 597, 603, 624, 629
VSAM records, size 54
VSAM recovery through ISPF panels 376
VSAM restore
  automatic during forward recovery 448
  IDCAMS delete/define 451
  management overview 71
  overview 448
  point-in-time restore 449
  procedure 457
  restore of non-RUV backups 450
  restore of SMS backups 451
VSAM sets
  deleting 212
  information gathering forms 662
  overview 63
  processing 199
  reports 430
  use in rule set 157
VSAM sphere recovery selection list 376
VSAM sphere recovery with ISPF panels 376
VSAM_DEFINITION variable 176, 714
176, 715
VSAM_FILE keyword
  on ADD VSAM_FILE command 147, 149
  on ARCHIVE ACCUM command 254
  on DELETE VSAM_FILE command 147, 150
  on PURGE REPOSITORY command 589
  on RECOVER command 482, 488, 494
  on REGISTER BACKUP_FILE command 355, 357, 359
  on REPORT command 425
  on RESTORE VSAM_FILE command 458
  on UPDATE VSAM_FILE command 147, 149
VSAM_FILE_MASK keyword 306
VSAM_GROUP keyword 444
  on ARCHIVE ACCUM command 254
  on PURGE REPOSITORY command 589
  on REPORT command 427
VSAM_GROUP_DEFINITION keyword
  on DELETE command 212
  on REPORT command 428
VSAM_IN keyword 292, 296, 440
VSAM_NAME_MODEL keyword 212, 305, 442, 446
  on REPORT command 430
176, 714
VSAM_RULE keyword 159, 614
VSAM_SET keyword 212
  on ADD RULE_SET command 192
  on REPORT command 430
<VSOFF> variable 180, 716
<VSON> variable 180, 716
W

WORK_DISK keyword 141
WORK_TAPE keyword 141
**X**

X37 support

<+1> keyword 133

<+1> variable 176, 177, 715

XBM subsystem 141

XBMID keyword 141, 431

**Z**

z/OS console commands for BWO 221
Notes