BMC Next Generation Technology Recover for DB2 for z/OS
Reference Manual

Supporting

Version 12.1.00 of BMC Next Generation Technology Recover for DB2 for z/OS
Version 12.1.00 of Administrative Assistant for DB2
Version 12.1.00 of BMC Database Administration for DB2
Version 12.1.00 of BMC Recovery Management for DB2
Version 12.1.00 of BMC Recovery for DB2

December 2016
Contacting BMC Software

Several methods are available for 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 2103 CITYWEST BLVD HOUSTON TX 77042-2827 USA</td>
<td>1 713 918 8800 or 1 800 841 2031</td>
<td>1 713 918 8000</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 1994-2016 BMC Software, Inc.

BMC, BMC Software, the BMC logo, the BMC Software logo, and other BMC marks are the exclusive properties of BMC Software, Inc. and are registered or may be registered in the U.S. and in other countries. BladeLogic and other BladeLogic marks are the exclusive properties of BladeLogic, Inc. and are registered or may be registered in the U.S. and in other countries. All other trademarks or registered trademarks are the property of their respective owners.

DB2, FlashCopy, IBM, IBM z, MVS, QMF, RACF, System z, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product 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 UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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, 2103 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
**Customer support**

**Support website**
You can obtain technical support from BMC 24 hours a day, 7 days a week at [http://www.bmc.com/support](http://www.bmc.com/support). 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 problems similar to yours and possible solutions
- Order or download product documentation
- Download products and maintenance
- Report a problem or ask a question
- Subscribe to receive proactive e-mail alerts
- 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 1 800 537 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 problem**
- **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
  - Messages from related software
License key and password information

If you have questions about your license key or password, contact Customer Support through one of the following methods:

- Send an e-mail message to customer_support@bmc.com. (In the Subject line, enter SupID:yourSupportContractID, such as SupID:12345.)
- In the United States and Canada, call 1 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
Contents

About this book 15
Related publications ...........................................................................................................15
Conventions .........................................................................................................................16
Syntax diagrams ..................................................................................................................16
Summary of changes ..........................................................................................................17
  Version 12.1.00 December 2016 ................................................................................18
  Version 11.2.00 May 2015 ........................................................................................22
  Version 11.1.00 June 2013 ........................................................................................24

Chapter 1 Overview of BMC Next Generation Technology Recover for DB2 for z/OS 31
  Advantages of NGT Recover ............................................................................................31
  Solution integration ............................................................................................................32
  NGT Recover features ........................................................................................................33
    Reduced elapsed time required to recover .................................................................33
    Better audit and recovery process control .................................................................34
    Expanded recovery capabilities .................................................................................35
    Recovery test capability ...............................................................................................37
    TRANSFORM and High-speed Structure Change ......................................................37
  NGT Recover operation .....................................................................................................38
    Using IBM DB2 resources ............................................................................................40
    Using NGT Recover with RECOVERY MANAGER ..................................................40
    Using NGT Recover with R+/CHANGE ACCUM .....................................................41
  NGT Recover limitations ....................................................................................................41
  Planning recovery ...............................................................................................................43
  Recovery strategies .............................................................................................................44
    Using LOGSCAN for estimates ....................................................................................44
    Using BACKOUT for point-in-time recovery without image copies .........................45
    Comparing BACKOUT performance .........................................................................46
    Using BACKOUT AUTO recovery ............................................................................47
    Using copies and logs for index recovery ....................................................................48
    Performing forward recovery without an image copy .................................................48
    Recovering with Instant Snapshot copies ....................................................................48
    Using parallel MERGE phases and log sorts ...............................................................49
    Using multitasking index key sorts and index rebuilds .............................................49
    Eliminating interim data sets for index keys ..............................................................50
Rebuilding nonpartitioned indexes on a partitioned table space ........................................... 50
Using resource selection ........................................................................................................ 50
Previewing recovery activity .................................................................................................. 51
Testing recovery with simulation ......................................................................................... 52
Recovering while activity continues ..................................................................................... 54
BMCXCOPY registrations ....................................................................................................... 54
Creating copies from recovery resources ............................................................................ 55
Overriding registered copies ............................................................................................... 56
Recovering LOB spaces with NGT Recover ......................................................................... 56
Automatic fallback without restarting ................................................................................. 57
Using incremental index image copies ............................................................................... 58
Using cabinet copies ............................................................................................................ 60
Using Online Consistent Copy ............................................................................................ 61
Using a system-level volume backup .................................................................................. 62
Migrating or importing data ................................................................................................. 63
Solving other problems with NGT Recover .......................................................................... 63

Chapter 2 Operational considerations

Operating environment ........................................................................................................... 65
  DB2 support ......................................................................................................................... 65
  System requirements ........................................................................................................... 65
  zIIPs ...................................................................................................................................... 66
  Setting the MEMLIMIT parameter ..................................................................................... 66
  Software requirements ......................................................................................................... 67
Authorization .......................................................................................................................... 68
  Authorization verification mechanisms .............................................................................. 68
  DB2 authority ...................................................................................................................... 68
  System authority ................................................................................................................ 69
  APF authority ..................................................................................................................... 70
Table space and index space status ....................................................................................... 70
  Initial table space and index space status .......................................................................... 71
  Restoring initial status ....................................................................................................... 74
Serialization and concurrency issues ..................................................................................... 74
  Concurrency with IBM DB2 utilities ................................................................................... 74
  Concurrency with other BMC utilities for DB2 ................................................................. 75
  Concurrency with other NGT Recover executions ............................................................. 76
Recovery using a non-DB2 backup ........................................................................................ 77
Recovery using cumulative incremental image copies ....................................................... 78
Checkpoints for NGT Recover restart .................................................................................. 78
  Checkpoint installation options ........................................................................................ 79
  Checkpoint override parameter ......................................................................................... 79
Chapter 3  NGT Recover syntax  83
OPTIONS ................................................................. 83
OUTPUT ................................................................. 83
RECOVER TABLESPACE, RECOVER INDEX, RECOVER INDEXSPACE ............. 84
REBUILD INDEX ...................................................... 84
RECOVER UNLOADKEYS ........................................... 85
RECOVER BUILDINDEX ............................................. 85
LOGSCAN ............................................................. 86
MIGRATE ............................................................... 86
IMPORT ................................................................. 86
Adding comments ...................................................... 87
Use of long names .................................................... 87
Support for Unicode names ........................................ 88
Using multiple commands ....................................... 88
  General restrictions for multiple commands ............... 89
  Restrictions for STOGROUP-defined objects ................. 90
  General information about SORTDEVT and SORTNUM .......... 90
  Restrictions for key work data sets, SKEYDDN, SORTNUM, and SORTDEVT ......... 91
Alphabetical listing of NGT Recover options ................. 92
OPTIONS command .................................................. 103
  OPTIONS syntax .................................................. 104
  OPTIONS option descriptions ................................ 107
OUTPUT command ................................................... 142
  OUTPUT syntax ..................................................... 143
  OUTPUT command dependencies and prohibitions ........ 145
  OUTPUT option descriptions .................................. 145
RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands .................................................. 151
  RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax ................................................................. 151
  RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE command dependencies and prohibitions ................ 164
  Additional RECOVER INDEX and RECOVER INDEXSPACE limitations .......... 165
  RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE option descriptions .................................................. 167
Copy data set ddname construction for JCL-allocated data sets .................. 214
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>REBUILD INDEX</td>
<td>215</td>
</tr>
<tr>
<td>REBUILD INDEX syntax</td>
<td>216</td>
</tr>
<tr>
<td>REBUILD INDEX options</td>
<td>218</td>
</tr>
<tr>
<td>RECOVER UNLOADKEYS</td>
<td>237</td>
</tr>
<tr>
<td>RECOVER UNLOADKEYS syntax</td>
<td>238</td>
</tr>
<tr>
<td>RECOVER UNLOADKEYS options</td>
<td>238</td>
</tr>
<tr>
<td>RECOVER BUILDINDEX</td>
<td>246</td>
</tr>
<tr>
<td>RECOVER BUILDINDEX syntax</td>
<td>247</td>
</tr>
<tr>
<td>RECOVER BUILDINDEX options</td>
<td>247</td>
</tr>
<tr>
<td>LOGSCAN</td>
<td>253</td>
</tr>
<tr>
<td>LOGSCAN syntax</td>
<td>254</td>
</tr>
<tr>
<td>LOGSCAN command dependencies and prohibitions</td>
<td>257</td>
</tr>
<tr>
<td>LOGSCAN options</td>
<td>258</td>
</tr>
<tr>
<td>MIGRATE</td>
<td>270</td>
</tr>
<tr>
<td>MIGRATE syntax</td>
<td>271</td>
</tr>
<tr>
<td>MIGRATE command dependencies and prohibitions</td>
<td>275</td>
</tr>
<tr>
<td>MIGRATE options</td>
<td>276</td>
</tr>
<tr>
<td>IMPORT</td>
<td>290</td>
</tr>
<tr>
<td>IMPORT syntax</td>
<td>291</td>
</tr>
<tr>
<td>IMPORT command dependencies and prohibitions</td>
<td>294</td>
</tr>
<tr>
<td>IMPORT options</td>
<td>295</td>
</tr>
</tbody>
</table>

Chapter 4 Building and running NGT Recover jobs 311

Building NGT Recover jobs 311
- JOB statement 312
- EXEC statement 312
- REGION parameter 312
- Utility parameters on the EXEC statement 313
- STEPLIB DD or JOBLIB statements 319
- NGT Recover data sets and NGT Recover DD statements 319
- Defining sort work data sets 329
- Determining sort work space requirements 330
- Determining unloaded keys file space requirements 333
- Specifying NGT Recover syntax 334

Running NGT Recover jobs 334
- Starting an NGT Recover job 335
- Restarting an NGT Recover job 335
- Displaying the status of an NGT Recover job 340
- Terminating an NGT Recover job 340
- Cleaning up the BMCUTIL and BMCSYNC tables 340
- Cleaning up after a RECOVER UNLOADKEYS job 341
Using MSGLEVEL to control NGT Recover output ....................................................341
  Sample output with MSGLEVEL(0) .................................................................342
  Sample output with MSGLEVEL(1) .................................................................345
  Sample output with MSGLEVEL(2) .................................................................349
Examples of NGT Recover operations ....................................................................358
  Multiple table space recovery and index rebuilds .............................................359
  Parallel index key sorts and multitasking index rebuilds ...................................361
  Multiple log sorts, MERGE phases, and parallel index key sorts with
  multitasking index rebuilds ................................................................................362
  Multiple table space and index recoveries ..........................................................364
  BACKOUT recovery .........................................................................................364
  Instant Snapshot recovery ................................................................................365

Chapter 5 Examples of NGT Recover jobs 367
Example 1: Previewing the recovery plan ...............................................................373
Example 2: Recovering to the current state ..........................................................378
  Example 2A: Using REBUILD INDEX .............................................................378
  Example 2B: Using RECOVER INDEX ............................................................383
Example 3: Creating and dynamically allocating output copies .........................397
Example 4: Recovering to a specified copy ..........................................................397
  Example 4A: Using TOCOPY with a named copy ............................................398
  Example 4B: Using TOCOPY LASTCOPY .........................................................398
Example 5: Recovering to a specific log point .......................................................400
  Example 5A Using the TOLOGPOINT option with RECOVER .......................401
  Example 5B: Using TOLOGPOINT and BACKOUT .........................................401
Example 6: Extracting nonpartitioned index keys ...............................................402
Example 7: Building a nonpartitioned index ........................................................402
Example 8: Recovering using tape-stacked data sets ..........................................403
  Example 8A: Making stacked image copies ......................................................403
  Example 8B: Recovering with stacked image copies ........................................404
Example 9: Recovering to a non-DB2 data set .......................................................404
  Directing recovery output ................................................................................405
Example 10: Overriding installation options .........................................................415
  Example 10A: Overriding several installation options ......................................415
  Example 10B: Overriding the IXRECP installation option ...............................416
Example 11: Recovering using a change accumulation file ....................................422
  Example 11A: Creating the change accumulation file ......................................422
  Example 11B: Using the change accumulation file ........................................422
Example 12: Using the MAINT parameter ............................................................423
Example 13: Using the LOGSCAN command .......................................................423
Example 14: Simulating recovery ........................................................................426
Example 15: Using MAXKSORT for parallel index rebuilds ........................................... 426
Example 16: Using MAXKSORT and recovering a table space .......................................... 434
Example 17: Using timestamp recovery ........................................................................ 435
Example 18: Using the MAXLSORT default value ............................................................ 436

Chapter 6          NGT Recover concepts 451
Recovery of partitioned table spaces and indexes ............................................................ 451
  Recovering multiple table space partitions in separate jobs ........................................ 451
  Recovering table space partitions and rebuilding a partitioned index in multiple jobs ........................................................................................................... 453
Rebuilding nonpartitioned indexes with single and multiple jobs .................................. 455
Recovering a partitioned table space and rebuilding a nonpartitioned index ................. 456
Recovering a table space and rebuilding a partitioned index and a nonpartitioned index ........................................................................................................... 458

Overview of the LOGSORT strategy ............................................................................ 458
Why is it faster to sort the log? ..................................................................................... 459
What are the restrictions when using LOGSORT? ....................................................... 460
Does LOGSORT have other advantages or disadvantages? ........................................ 460

Using LOGONLY or LOGAPPLY ONLY with point-in-time recoveries ......................... 461

Overview of the NOWORKDDN strategy ..................................................................... 463

Overview of the UNLOADKEYS and BUILDINDEX strategy .................................... 463

Using SHRLEVEL CHANGE full copies for recovery .................................................... 464
  Attempting to recover to a SHRLEVEL CHANGE copy ........................................... 465

Fallback recovery with NGT Recover .......................................................................... 465

Strategies for point-in-time recovery ......................................................................... 466
  Using BACKOUT ........................................................................................................ 466
  Using Instant Snapshots ............................................................................................ 469

Changing SHAREOPTIONS for striped or extended format data ............................... 473

Working with ASCII and Unicode data ...................................................................... 473

Managing sort performance ....................................................................................... 474
  Recommendations for a large number of concurrent sorts .................................... 475
  Sort file size estimation ............................................................................................ 475
  NGT Recover sort parameters .................................................................................. 477
  Allocation of sort work space .................................................................................. 483

Strategies for using copies ........................................................................................ 484
  Image copy data set contents .................................................................................. 484
  Cabinet copy data set contents ............................................................................... 484
  Using copies containing multiple data sets ............................................................. 485
  Using data sets stacked on tape ............................................................................... 485
  Using image copies and log records for indexes ..................................................... 486
Using R+/CHANGE ACCUM with NGT Recover .................................................... 487
Using change accumulation files for multiple objects ........................................ 488
Using separate change accumulation files for separate data sets .................... 489
Planning recovery resources .............................................................................. 489
Tips on grouping recovery resources ................................................................. 490
Reading multiple log files concurrently .............................................................. 490
Allocating output image copy data sets dynamically ........................................... 491
Using copy data set output descriptors .............................................................. 492
GDGs and symbolic variables in data set name construction ............................... 493
Stacking copies on tape .................................................................................... 498
Automatic sizing of output copies dynamically allocated on DASD ................... 499
Using NGT Recover with data sharing .............................................................. 500
Forward recovery of versioned index spaces ..................................................... 500
Supporting real-time statistics in NGT Recover .................................................. 501
Recovering altered spaces ................................................................................ 502
Added partitions ............................................................................................... 503
Rotated partitions ......................................................................................... 503
Inserted partitions ......................................................................................... 504
Dropped partitions ......................................................................................... 504
Changed limit keys .......................................................................................... 504
Materialized pending changes ....................................................................... 505
Recovering encrypted copies ........................................................................... 506
Requirements for encryption ........................................................................... 507
Key data set .................................................................................................... 507
Registration of copies ..................................................................................... 512
Handling DB2 versioning information ............................................................... 513
Recovering cloned objects .............................................................................. 514
Recovering XML objects ................................................................................ 516
Recovering compressed indexes ..................................................................... 517
Using BMC RECOVERY MANAGER groups .................................................... 518
Using MIGRATE and IMPORT for data movement ......................................... 520
NGT Copy EXPORT file .................................................................................. 522
NGT Recover MIGRATE and IMPORT commands ........................................ 522
Limitations ...................................................................................................... 523

Chapter 7   Recovering a dropped table space or table 525
Overview of recovering dropped table spaces or tables .................................... 525
Example 1: Recovering a dropped table space when image copies are available .. 526
Example 2: Recovering a dropped table in a segmented table space containing multiple tables ................................................................. 531
Example 3: Recovering a dropped table from a simple table space .................... 535
About this book

This book contains detailed information about the associated product or products. This preface explains the special conventions that the book uses, and how to access related publications.

If applicable, the preface also summarizes the major changes included in the latest release of the product.

Related publications

From the BMC Support Central website, you can use the following methods to access related publications that support your product or solution:


- View Quick Course videos (short overviews of selected product concepts, tasks, or features), which are available from the following locations:
  
  — Documentation Center (primary center and secured center)
  
  — Support Central (at http://www.bmc.com/support/mainframe-demonstrations)
  
  — BMC Mainframe YouTube channel (https://www.youtube.com/user/BMCSoftwareMainframe)


Products with online interfaces also offer online Help via the F1 key or, for graphical user interfaces (GUIs), via a Help button.
**Tip**

If you prefer hardcopy documentation, you can order it from your BMC sales representative or from Support Central. Also, from Support Central you can subscribe to receive proactive e-mail alerts when BMC issues notices.

---

**Conventions**

This document uses the following special conventions:

- All syntax, operating system terms, and literal examples are presented in this typeface.

- Variable text in path names, system messages, or syntax is displayed in italic text:
  
  `testsys/instance/fileName`

- Menu sequences use a symbol to convey the sequence. For example, **Actions => Create Test** instructs you to choose the **Create Test** command from the **Actions** menu.

---

**Syntax diagrams**

The following figure shows the standard format for syntax diagrams:
The following example illustrates the syntax for a hypothetical DELETE statement. Because the FROM keyword, alias variable, and WHERE clause are optional, they appear below the main command line. In contrast, the tableName variable appears on the command line because the table name is required. If the statement includes a WHERE clause, the clause must contain a search condition or a CURRENT OF clause. (The searchCondition variable appears on the main line for the WHERE clause, indicating that this choice is required.)

The following guidelines provide additional information about syntax diagrams:

- Read diagrams from left to right and from top to bottom.
- A recursive (left-pointing) arrow above a stack indicates that you may choose more than one item in the stack.
- An underlined item is a default option.
- If a diagram shows punctuation marks, parentheses, or similar symbols, you must enter them as part of the syntax.
- In general, IBM commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum required portion might be shown in uppercase letters, with the remainder in lowercase (for example, CANcel).
- The following conventions apply to variables in syntax diagrams:
  - Variables are typically displayed in lowercase letters and are always italicized.
  - If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words (for example, databaseName).

Summary of changes

This section summarizes changes to the functionality of the product, listing the changes by product version and release date.

The summary includes enhancements to the product and any major changes to the documentation.
Version 12.1.00 December 2016

This release of NGT Recover includes the following product enhancements and changes.

RECOVER PLUS for DB2 name change

Starting with this release, the name of the RECOVER PLUS for DB2 product has changed to the BMC Next Generation Technology Recover for DB2 for z/OS (or NGT Recover) product.

COPY PLUS for DB2 name change

Starting with this release, the name of the COPY PLUS for DB2 product has changed to the BMC Next Generation Technology Copy for DB2 for z/OS (or NGT Copy) product.

Support for IBM DB2 Version 12

This release adds support for DB2 Version 12.

This release supports:

- DB2 Version 12
- DB2 Version 11
- DB2 Version 10 in new-function mode (NFM) only

This includes support for:

- Table spaces defined with PAGENUM RELATIVE
- The recovery of compressed LOBs
- Active logs with a capacity greater than 4 GB
- Column level CCSID specification
- The fast insert algorithm (INSERT ALGORITHM 2)

Support for R+/CHANGE ACCUM on IBM DB2 Version 12

This release of NGT Recover supports R+/CHANGE ACCUM on DB2 Version 12.

NGT Recover can read change accumulation files created in R+/CHANGE ACCUM version 11.1 or later. NGT Recover ignores change accumulation files created by earlier versions of R+/CHANGE ACCUM and uses the DB2 logs instead.
**Additional zIIP offload**

Starting with this release, NGT Recover offloads substantially more rebuild index processing to the IBM z Integrated Information Processor (zIIP). This can reduce central processor (CP) time by 50% or more.

**Performance improvements for REBUILD INDEX**

This release introduces a number of significant optimizations to the REBUILD INDEX code that substantially reduce processing time and the time it takes to rebuild indexes.

**Recovery performance improvements**

Starting with this release, NGT Recover can run:
- The log input phase in a subtask
- Snap, reset space, restore, and merge phases in parallel with log input
- Snap phases for recovery of a multi-data set nonpartitioned index (NPI) in parallel in subtasks

**Support for ARCHIVE tables**

This release adds the AUX ARCHIVE option to support the recovery of ARCHIVE tables.

**Performance improvements for IMPORT**

Starting with this release, NGT Recover can skip unchanged index partitions when using the IMPORT command.

**Requirement for consistent use of INDEP OUTSPACE**

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it.

If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

**Improved data set preallocation**

This release increases the number of data spaces that NGT Recover can preallocate before the main task selects them for processing. This enhancement can reduce the
total elapsed processing time, especially when you have many partitions or many small spaces.

**Recovery of archive tables**

You can now use the ARCHIVE keyword after the AUX option to recover archive tables.

**Forward recovery of indexes to a timestamp**

This release supports the ability of NGT Recover to perform a forward recovery of indexes to a timestamp by using the RECOVER INDEX command with OPTION RECOVERYPOINT.

**Recovery to a point in time before materializing changes**

Starting with this release, you can recover a partition-by-growth universal table space (PBG UTS) to a point in time (PIT) that precedes the materialization of certain table space attributes (SEGSIZE, DSSIZE, PGSIZE, and MEMBER CLUSTER).

See “Materialized pending changes” on page 505.

**REBUILD INDEX SHRLEVEL CHANGE supports indexes in rebuild pending status**

You can now use the SHRLEVEL CHANGE option for indexes in RBDP or PSRBD status and VCAT-defined table spaces.

**Removal of restrictions on recovery simulation**

This release now supports recovery simulation of:

- Recovery from Instant Snapshot copies
- BACKOUT recovery
- LOGONLY or LOGAPPLY SCANONLY
- INLINE image copies

**Access to the SYSLGRNX table if you are using IBM DB2 Version 10**

NGT Recover no longer supports indirect access to the SYSIBM.SYSLGRNX table but supports direct access as follows:
If you are using DB2 Version 10, ensure that CATMAINT has been executed for IBM APARs PM35190 and PM55333. These APARs define the SYSLGRNX table in the DB2 catalog. With the APARs applied, NGT Recover can access SYSLGRNX directly via the DB2 catalog.

If you are using DB2 Version 11 or later, you have direct access to SYSLGRNX; starting with Version 11, DB2 automatically defines SYSLGRNX in the DB2 catalog.

**SCOPE, RECOVERSCOPE, and REBUILDSCOPE options**

This release adds the RECOVERSCOPE and REBUILDSCOPE options to the OPTIONS command, and the SCOPE option to the RECOVER and REBUILD commands.

The following SCOPE options enable NGT Recover to bypass objects based on specified criteria:

- **SCOPE UPDATED** bypasses spaces that have not changed since a specified recovery point. This option is available for all RECOVER and SIMRCVR commands. NGT Recover uses SCOPE UPDATED by default for local site forward recovery to a point in time.

- **SCOPE ALL** recovers all specified spaces.

- **SCOPE STATUS (status1, status2,...)** selects objects for processing based on their specified statuses. This option is available for RECOVER TABLESPACE, RECOVER INDEX, and RECOVER OBJECTSET commands.

- **SCOPE PENDING** processes only those indexes in a RBDP, PSRBD, RBDP*, or RECP status. This option is available for RECOVER INDEX and REBUILD INDEX commands.

The RECOVERSCOPE and REBUILDSCOPE option settings are the same as the SCOPE option settings.

**New &PART4 symbolic variable**

You can use a new symbolic variable, &PART4, to generate partitions for data set allocation. You can use &PART4 for any data set.

See “Using symbolic variables” on page 494.
Substrings of symbolic variables

You can use the following substrings of symbolic variables in data set names specified by OUTCOPYDSN, RECOVERYDSN, and INCOPY MODEL dataSetName:

- &DB(s,l)
- &TS(s,l)
- &USERID(s,l)
- &USER(s,l)
- &UID(s,l)

The s variable represents the substring and the l variable represents the length.

See “Using symbolic variables” on page 494.

Subgroup name attachment

NGT Recover now allows you to specify a subgroup name for the IBM DB2 subsystem identifier (ssid).

Version 11.2.00 May 2015

This release of RECOVER PLUS includes the following product enhancements and changes:

- End of support for DB2 Version 9
  Starting with this release, RECOVER PLUS does not support IBM DB2 Version 9. Earlier releases will continue to support Version 9. In addition, RECOVER PLUS supports DB2 Version 10 only in new-function mode.

- Support tagged copies for migration
  In the previous release, when EXPORT was run to identify copies for migration, it could be confusing as to which copies were exported, especially if the copies were made in multiple jobs. This release allows RECOVER PLUS to "tag" output copies to simplify selecting copies for migration.
  See TAG tagName in “OUTCOPY specification” on page 193.

- Migrate XML spaces
  This release adds support for migrating XML spaces using image copies.

- Migrate statistics
  This release adds support for migrating statistics. See STATS in the IMPORT COMMAND.
- Allow OUTCOPY ONLY using inline copies
  This release adds support for making output copies using inline input copies.

- Allow OUTCOPY ONLY using Instant Snapshots
  This release adds support for making output copies using Instant Snapshot input copies.

- REBUILD INDEX SHRLEVEL CHANGE
  Starting with this release, RECOVER PLUS supports rebuilding indexes with SHRLEVEL CHANGE. See “SHRLEVEL” on page 234 in the REBUILD INDEX command.

- Allow incremental copies with Instant Snapshots
  This release adds support for using incremental copies based on full Instant Snapshot copies for recovery.

- Online Consistent Copies for LOBs
  Starting with this release, RECOVER PLUS can recover LOB spaces using Online Consistent Copies.

- Support for CHANGE ACCUM on DB2 Version 11
  This release of RECOVER PLUS supports CHANGE ACCUM on DB2 Version 11.

- Recover individual spaces using system backups
  This release adds support for recovering individual spaces using system backups. See “RESOURCE SELECTION” on page 121.

- Instantiate data set for DEFINE NO
  When migrating data to a space that is DEFINE NO, starting with this release RECOVER PLUS will update the DB2 catalog and create the cluster if it does not exist.

- Migration of Partition by Growth (PBG) spaces with more than 1 data set
  Using the MIGRATE command, if the target space has more than 1 data set, RECOVER PLUS will automatically add the data sets to the target system.

- Support for not indexing keys that are NULL
  Starting with this release, RECOVER PLUS can omit null keys during REBUILD INDEX.

- Support RBA/LRSN conversion during REBUILD INDEX
  Starting with this release, RECOVER PLUS can convert the RBA/LRSN format (basic-extended) during REBUILD INDEX. See “RBALRSN_CONVERSION” on page 237.
- Support RBA/LRSN conversion for point-in-time recovery
  Starting with this release, RECOVER PLUS allows recovery to a point in time where the RBA/LRSN format for the partition is different than the current RBA/LRSN format.

- Recover to a point in time before DROP COLUMN
  This release of RECOVER PLUS checks for DROP COLUMNs and disallows recovery to a point in time before a DROP COLUMN.

- Recover to a point in time before materializing changes
  Starting with this release, recovery to a point in time before certain materializing changes (e.g. DSSIZE, PGXIZE, and SEGSIZE) is allowed. See “Materialized pending changes” on page 505.

- Allow INDEP OUTSPACE with OBJECTSET
  Starting with this release, RECOVER PLUS allows the use of INDEP OUTSPACE with the RECOVER and REBUILD commands when an object list is specified via OBJECTSET syntax.

- New RICHK option
  If RICHK YES is specified and a partial recovery is performed on space(s) with referential integrity constraints, RECOVER PLUS will set check pending on related spaces that were not recovered. See RICHK in the OPTIONS (“RICHK” on page 116), IMPORT (“TABLESPACE specification” on page 298), and RECOVER (Table space specification on page 168) commands, and as an installation option “RICHK=NO” on page 591.

- New AUX ALL option
  In previous releases, AUX YES would recover history tables as well as the auxiliary spaces. This release separates history tables using the HISTORY option and adds AUX ALL to recover all of the auxiliary spaces. See “AUX” on page 117.

- Support for Persistent Read Only (PRO) and read-or-replication-only (RREPL) statuses
  This release adds support for PRO and RREPL statuses.

### Version 11.1.00 June 2013

This release of RECOVER PLUS includes the following product enhancements and changes:

End of support for DB2 Version 8
Starting with this release, RECOVER PLUS does not support IBM DB2 Version 8. Earlier releases will continue to support Version 8.

**Extended RBA and LRSN support**

All RECOVER PLUS output has been converted to show 10-byte RBAs and LRSNs. The BMC utilities database has also changed to support extended RBAs and LRSNs.

**New IMPORT and MIGRATE commands for migrating data**

New IMPORT and MIGRATE commands allow you to move data from one or more table spaces to another via a migration file that the COPY PLUS EXPORT command creates. The IMPORT command uses copies as input; the MIGRATE command uses copies and log. For more information, see the MIGRATE command description and the IMPORT command description.

---

**Note**

This feature requires one of the following valid passwords:

- Recovery Management solution password
- Database Administration solution password

The Database Administration (DAD) password allows RECOVER PLUS to run within a worklist, but does not include full-function RECOVER PLUS.

This release also includes the following changes:

- Adds new EXPSSID, EXPSLRSN, and EXPTLRSN columns to the BMCXCOPY table to accommodate the EXPORT, MIGRATE, and IMPORT commands
- Adds ICTYPE column value \( m \), and COPY_TYPE column values \( X \) and \( I \) to the BMCXCOPY table

In addition to the references above, see the following resources for more information:

- "Migrating or importing data"
- EXPORT command description in the *COPY PLUS for DB2 Reference Manual*
- "Moving data with a migration file," in *Recovery Management for DB2 User Guide*

**Transformation restrictions removed**

Transformation restrictions have been removed from High-speed Structure Change, as follows:
- Multi-table table space transformations are now allowed. Previously, the source table space could contain only one table.

- BRF to RRF and RRF to BRF transformations are now allowed. Previously, the source and target table space had to have the same row format. You could not transform a table space with basic row format (BRF) into a table space that had reordered row format (RRF), or vice versa.

  **Note**
  The rows are not transformed, but the DB2 catalog is changed to reflect the row format from the source.

- Transformations to and from MEMBER CLUSTER are now allowed. Previously, Recovery Management was unable to handle transformations related to the MEMBER CLUSTER attribute.


**Additional DSSIZE values**

The following values are now valid for DSSIZE:

- 128G
- 256G

**Estimation improvements**

Estimation has been improved to better account for current hardware and multitasking. Additionally, the following new installation options accommodate estimation:

- DISKIORATE specifies the number of megabytes per second that RECOVER PLUS reads from disk. The default value is 100.

- CPUMIPS specifies the rate at which a CPU executes instructions, in millions of instructions per second (MIPS). The default value is 200.

Many of the performance factors used in estimation are derived from DISKIORATE and CPUMIPS.

Estimation requires a Recovery Management for DB2 solution password. For more information, see "Recovery simulation and estimation," in the *Recovery Management for DB2 User Guide*.

**Use of symbolics to INCOPY MODEL dataSetName syntax**
The data set specified with INCOPY MODEL syntax supports the use of symbolics.

**New symbolic variables**

RECOVER PLUS adds the &UNIQ (or &UQ) symbolic variable to generate unique data set names. For example, you can use this variable with the DSNAME parameter on the OUTPUT command.

If you use &UNIQ, RECOVER PLUS generates a 1- to 8-character value that is based on the system clock. The first character is always an uppercase letter. Each remaining character is either an uppercase letter or a numeral from 0 through 9.

RECOVER PLUS also adds the &PART5 symbolic variable that you can use for any data set. RECOVER PLUS generates 5-character partition numbers as follows:

- Partition 1 = 00001
- Partition 10 = 00010
- Partition 100 = 00100
- Partition 1000 = 01000
- Nonpartitioned = 00000

**Allow INCOPY with OBJECTSET**

The restrictions that did not allow INCOPY with OBJECTSET have been removed.

**Automatic sizing of dynamically allocated output copies on DASD**

RECOVER PLUS now automatically sizes dynamically allocated output copies on DASD when the primary allocation quantity is not specified by the SPACE option.

To accomplish automatic sizing, RECOVER PLUS adds the MAXPRIM option to the OUTPUT command. MAXPRIM lets you set a maximum amount of disk space (in the units specified by SPACE) to allocate as primary space. Valid values are 0 through 65535. A nonzero value for MAXPRIM establishes an upper limit for primary space allocation; 0 specifies no limit.

**New EXCLUDE clause for OBJECTSET**

For the RECOVER and REBUILD INDEX commands, RECOVER PLUS adds the EXCLUDE option to the OBJECTSET specification.

**Support for VSAM data sets on nonsnappable disks**

When you use the new installation option SNAP, RECOVER PLUS can now recover VSAM copies, even if the data set is not on a snappable disk. SNAP is also available on the OPTIONS command. When the SNAP value is VSAM and the data set is not on a snappable disk, RECOVER PLUS reads a VSAM data set using the INCOPY
FULL SNAPSHOT DSNAME *dataSetName* syntax, and can also recover using a VSAM data set registered in BMCXCOPY or SYSCOPY.

Additionally, RECOVER PLUS adds the FCPPRC installation option to control what happens if you specify a value of VSAM with SNAP and the data sets are on a disk that is capable of IBM FlashCopy.

**New AFRPRnnn data sets to eliminate deferred messaging**

This release removes deferred messaging for subtasks. Now, each execution phase has its own AFRPRnnn print file, and messages are displayed in the print file as soon as they are issued. The maximum number of print files allowed by RECOVER PLUS is calculated by the following formula:

\[
\text{value of MAXKSORT} + \text{value of MAXLSORT} + 1 = \text{maximum number of AFRPR nnn files}
\]

The print files are named AFRPR001, AFRPR002, AFRPR003, and so on. AFRPRINT now displays the name of the AFRPR nnn file used by each phase for each object.

**New USELOGS installation option**

The new USELOGS installation option lets you specify the order in which to read active and archive log files. USELOGS=(ACT1,ACT2,ARC1,ARC2) is the default value.

You can override this installation option by using the RESOURCE SELECTION LOGS syntax on the OPTION command.

**Better handling of partition-by-growth migrations**

This release provides better handling for partition-by-growth (PBG) migrations. This release removes the warning against PBG migrations that have a different number of source and target data sets.

**Recovery of an individual space from a system-level volume backup**

To recover an individual space from a system-level volume backup, you can use COPY IMAGECOPY in the BMC COPY PLUS product to make a standard copy of the space. Then use RECOVER PLUS to recover the space from the standard copy.

**Documentation changes**

This release includes the following documentation changes:

- The Point-in-time recovery specification has been renamed to the TORBA/TOLOGPOINT specification in "RECOVER PLUS syntax" and in "RECOVER PLUS syntax diagrams."
■ All messages are now available in the BMC Documentation Center, which is accessible from the BMC Support Central site (http://www.bmc.com/support). A separate messages manual is no longer available.

■ Installation and configuration information is now located in the following books:

— *Installation System User Guide*

— *BMC Products and Solutions for DB2 Configuration Guide*
Overview of BMC Next Generation Technology Recover for DB2 for z/OS

This chapter describes the features and advantages of using the BMC Next Generation Technology Recover for DB2 for z/OS product.

Advantages of NGT Recover

The BMC Next Generation Technology Recover for DB2 for z/OS product (NGT Recover) provides an alternative to the IBM DB2 RECOVER utility in many functional areas. NGT Recover is a batch utility that runs outside the IBM DB2 subsystem and provides faster execution through advanced I/O techniques and the use of alternate recovery strategies. NGT Recover includes the following capabilities:

- Ability to create an image copy in parallel with the recovery
- Ability to create SHRLEVEL REFERENCE full copies without any impact to the real spaces (OUTCOPY ONLY option)
- Dynamic allocation of the output image copies
- BACKOUT recovery
- Recovery that uses Instant Snapshot copies made by the BMC Next Generation Technology Copy for DB2 for z/OS product (NGT Copy)
- Automatic fallback
- Recovery simulation, when you use the BMC Recovery Management for DB2 solution
- Consistent recovery to any point in time without a pre-established quiesce point, when you use the BMC Recovery Management for DB2 solution
The ability to recover a large number of data sets with a single data set allocation and deallocation when you use a cabinet copy, which you can create if you have a password for a supported solution.

NGT Recover also supports:

- Data sharing
- LARGE table spaces
- Inline copies
- ASCII data
- UNICODE data
- LOB table spaces
- XML table spaces and index spaces (UTSs)
- Universal table spaces
- Clone spaces
- Encrypted copies

NGT Recover offers enhanced concurrency when accessing DB2 resources and provides additional options to improve processing efficiency. The ability of NGT Recover to analyze and report planned recovery activity provides a high degree of confidence in the predictability of the recovery process.

Consequently, NGT Recover gives you more robust options than DB2 RECOVER for meeting aggressive customer demands for service-level agreements related to down time.

**Solution integration**

NGT Recover is a component of the BMC Recovery Management for DB2 solution. This solution integrates the features of the following BMC products and technologies:

- BMC Recovery Management for DB2
- BMC Next Generation Technology Recover for DB2 for z/OS
- R+/CHANGE ACCUM
- BMC Next Generation Technology Copy for DB2 for z/OS
- Log Master for DB2 with High-speed Apply Engine
- SNAPSHOT UPGRADE FEATURE (SUF), which is a licensed component of the EXTENDED BUFFER MANAGER (XBM) for DB2 product
- BMCSORT technology
- DB2 Solution Common Code (SCC) technology (a set of common components that several BMC DB2 products use)
Customers who acquire this solution benefit from all features of these products and technologies, as well as additional features that are available when one Recovery Management component can rely on the presence of all others. For more information, see the *Recovery Management for DB2 User Guide*.

NGT Recover is also a component of the Administrative Assistant for DB2 solution and the Database Administration for DB2 solution.

**NGT Recover features**

NGT Recover offers significant features and benefits.

**Reduced elapsed time required to recover**

NGT Recover offers several features that reduce the elapsed time that is required for recovery. For example, NGT Recover can invoke a MERGE routine that concurrently uses a full image copy, incremental copies, a change accumulation file, and sorted log to efficiently re-create a current table space image. Then, the MERGE routine unloads index keys for a rebuild. The MERGE concept is exclusive to NGT Recover and eliminates the need to read the table space page multiple times.

To further enhance the MERGE concept, NGT Recover can run multiple log sorts and parallel MERGE phases in subtasks. You use the MAXLSORT option in the installation options or on the OPTIONS command to specify the maximum number of log sorts that can run concurrently. MAXLSORT also determines the number of MERGE/RESTORE/SNAP phases that can run in parallel, whether or not log records are processed.

NGT Recover can perform multitasking index rebuilds in which multiple index key sorts and multiple index rebuilds are executed in parallel subtasks. You can define the level of concurrency to improve recovery performance by using the MAXKSORT option in the installation options or on the OPTIONS command. NGT Recover can extract keys from partitions in parallel if the partitions are not being recovered. You can also use the KSORTSHARE option, available as an installation option or on the OPTIONS command, to improve index recovery. KSORTSHARE specifies if key sorts are shared among NGT Recover table space recoveries (MERGE phases) running in parallel.

NGT Recover can use the RECOVER UNLOADKEYS and RECOVER BUILDINDEX commands to rebuild a nonpartitioned index in a two-step process. RECOVER UNLOADKEYS can run in several jobs to extract and sort the keys greatly reducing overall elapsed time, work data set requirements, or both. RECOVER BUILDINDEX uses the keys from RECOVER UNLOADKEYS to build the index.
NGT Recover considers all requests in the SYSIN data set and uses the most efficient technique for recovering all specified objects. This optimization process allows NGT Recover to achieve efficiencies such as extracting keys while a table space is written, detecting stacked tape inputs, and automatically using the inputs in order.

NGT Recover can use copies and log records to recover indexes, avoiding costly key sorts. Indexes can also be recovered with log only after being restored outside of DB2.

By using the BACKOUT option, NGT Recover can recover to a point in time without image copies by using the table spaces, index spaces, and log records to return to a prior state. This process reduces resource consumption in several ways:

- Only the log between the point in time specified and the current log point is read and processed, which may substantially reduce log processing.

- In addition, the spaces are read for log backout processing, and image copy processing is completely eliminated.

- If indexes are not rebuilt and output image copies are not requested, only the pages that are updated by log records are read and written, which can greatly reduce I/Os and speed up the recovery.

NGT Recover can recover by using Instant Snapshot copies, which are data set level copies made by the BMC NGT Copy product with the BMC EXTENDED BUFFER MANAGER (XBM) product. Instant Snapshots are non-standard, point-in-time copies that are made on intelligent storage devices.

When you use NGT Recover as part of the BMC Recovery Management for DB2 solution, NGT Recover can recover DB2 table spaces and indexes to any point in time and resolve inflight transactions at that point.

**Better audit and recovery process control**

NGT Recover provides early recall of migrated data sets and the ability to make up to four copies of the recovered space and to register one or more of them.

When you use NGT Recover for a point-in-time recovery, NGT Recover allows generic keywords, such as TOCOPY LASTCOPY or TOLOGPOINT LASTQUIESCE, to identify the target copy or log point.

NGT Recover can generate a list of archive, image copy, and change accumulation file tapes required for recovery. A report that lists all volumes of tapes and cartridges that are dynamically allocated during recovery is also available.
NGT Recover provides a command to scan logs and provide estimates to help plan your recovery.

The NGT Recover SIMULATE option, which is available if you have a valid BMC Recovery Management for DB2 solution password, allows you to exercise all recovery resources. This option performs all non-destructive operations of the recovery to validate that all image copies, logs, and change accumulation files are available and usable.

## Expanded recovery capabilities

NGT Recover provides expanded recovery capabilities that enable you to recover from a dropped table space, create image copies without reading the table space or index space, and migrate data to other DB2 systems. With NGT Recover, you can perform object ID translation, use unregistered input copies, and run recovery without SYSLGRNX entries.

### Object ID translation

You can use the NGT Recover OBIDXLAT option to translate internal DB2 object IDs (OBIDs) while processing. This feature supports the NGT Recover drop recovery function to allow recovery of dropped table spaces and allows translation during data migration. Improved migration checking when the INCOPY option is used detects incompatible migrations. Some examples of incompatible migrations are:

- Migration of a partitioned table space into a nonpartitioned table space
- Migration into a table space with a different page size
- Migration of a segmented table space into a table space with a different segment size

NGT Recover translates OBIDs and resets log point values in the pages, but in all other respects the internal structure of the source, target table spaces, and indexes must be the same.

When you use Instant Snapshots for migration and specify the OBIDXLAT option, NGT Recover automatically detects if the OBIDs have not changed and avoids the overhead of reading and writing all of the pages.

For more information about the drop recovery function, see “Recovering a dropped table space or table” on page 525.

With the Copy Migration feature of the Recovery Management solution, NGT Recover will obtain the object ID values from a migration file and either the source or target catalog, simplifying the migration process. For more information, see the Recovery Management for DB2 User Guide.
Optional use of non-registered input copies

The INCOPY option allows you to specify the following types of copies as input:
- Copies that have been dropped from SYSIBM.SYSCOPY
- Copies made with DSN1COPY
- Instant Snapshot copies that are not registered in BMCXCOPY
- Copies that were made on another subsystem (or data sharing group)

Using non-registered input copies supports the following important functions of NGT Recover:
- Migration of data
- Recovery of dropped table spaces

With the Copy Migration feature of the Recovery Management solution, NGT Recover will obtain the input image copy name from a migration file or the source catalog, simplifying the migration process. For more information, see the *Recovery Management for DB2 User Guide*.

Optional use of non-registered table space reset events

Certain events that are recorded in SYSCOPY allow recovery of table spaces to be performed solely from the log. These events include LOAD LOG YES with REPLACE or REORG LOG YES. If these events no longer exist in SYSCOPY because of a table space being dropped or modified, the INLOG RBA/INLOG LOGPOINT option can be used to specify the starting point of a log-only recovery. The table space will be reset, and the log will be used to rebuild the table space. This feature does not apply to indexes.

Running without SYSLGRNX records

The NOSYSLGRNG option allows NGT Recover to scan the log without referencing the DB2 SYSLGRNX table when processing a table space. The log is scanned from the beginning log point, such as the log point of the last copy, to TORBA/TOLOGPOINT or current. For indexes with the COPY NO attribute, index log records are always read without referencing the SYSLGRNX table.

This feature supports the NGT Recover drop recovery function. This feature is also useful if you have a damaged SYSLGRNX table.

OUTCOPY ONLY

You can use the OUTCOPY ONLY option to build image copies without reading the table or index space or interfering with DB2 update access in any way. This feature allows you to avoid DASD contention and supports data migration.
Recovery test capability

The NGT Recover simulation feature allows you to test DB2 recovery without performing any destructive operations or affecting the availability of the target table space or index. The BMC Recovery Management for DB2 solution employs this feature to simulate a Disaster Recovery test.

TRANSFORM and High-speed Structure Change

Use the NGT Recover TRANSFORM syntax to specify that you want to do a DDL transformation.

Note

TRANSFORM is part of the High-speed Structure Change process that includes the following methods:

- The SHRLEVEL REFERENCE method that requires either a Recovery Management password, or BMC Next Generation Technology Copy for DB2 for z/OS and EXTENDED BUFFER MANAGER passwords or NGT Copy and SNAPSHOT UPGRADE FEATURE passwords for the creation of the initial image copies as well as an NGT Recover password for the TRANSFORM operation.
- The SHRLEVEL CHANGE method that requires a Recovery Management password because of the use of Online Consistent Copies.

The High-speed Structure Change process allows you to perform the following transformations:

- Simple table space to partition-by-growth (PBG) table space
- Non-large partitioned table space to large partitioned table space
- Range-partitioned table space to range-partitioned (PBR) universal table space
- Range-partitioned table space to partition-by-growth universal table space
- Multi-table table space to partition-by-growth universal table space

The High-speed Structure Change process also allows you to transform your indexes, avoiding costly sort CPU that would be required to rebuild the index.

At a more granular level, you can use the High-speed Structure Change process to:

- Change the segment size (SEGSIZE) of a table space
- Transform a non-segmented table space into a segmented one
- Change the data set size (DSSIZE or PIECESIZE) of your table space or index
- Change the compression attribute of your index
- Add or remove the MEMBER CLUSTER attribute
NGT Recover performs the page transformations and uses the TRANSFORM option (Table space specification on page 168), the DSIZE option (“INCOPY specification” on page 182), and the PIECESIZE option (“INCOPY specification” on page 182).

For more information, see the Recovery Management for DB2 User Guide.

NGT Recover operation

To accomplish recovery tasks, NGT Recover offers the following resources:

- Uses the same DB2 resources that are used by the DB2 RECOVER utility
- Uses any available change accumulation files as input to forward recovery
- Uses index image copies that are made by NGT Copy, the IBM DB2 COPY utility, NGT Recover, or DSN1COPY
- Uses Instant Snapshot copies that are made by NGT Copy
- Uses encrypted copies that are made by NGT Copy
- Uses cabinet copies that are created by NGT Copy as part of the Recovery Management solution
- Uses Online Consistent Copies, which you can create if you have the Recovery Management solution
- Supports a wide range of recovery strategies by using a variety of commands and options

During a recovery, NGT Recover automatically optimizes recovery processing by performing the following tasks:

- Optimizing block sizes
- Extracting index keys, when possible and requested, during table space recovery
- Detecting and ordering activities for the proper positioning of stacked tape data sets, change accumulation files, and input and output copies containing all partitions of a partitioned space
- Using a subtask to preallocate and open VSAM data sets required during recovery
Automatically falling back, without restarting, to a previous copy or Instant Snapshot, if an invalid or unavailable image copy or Instant Snapshot is encountered during recovery.

Log data sets and table space data sets are dynamically allocated to the job. NGT Recover also dynamically allocates the input image copy data sets and required change accumulation files. The target table and index spaces are identified through IBM-compatible syntax, and the target DB2 subsystem is identified in the EXEC statement utility parameters. As an option, NGT Recover provides support for dynamic allocation of output image copies.

NGT Recover parses all NGT Recover command statements before initiating recovery. NGT Recover constructs a recovery plan and produces a report on this plan. Other options allow control over other aspects of the recovery. For example, you can specify the device type (SORTDEVT) and number of temporary data sets (SORTNUM) for sorting, or you can specify the MAXKSORT option for multitasking index key sorts and index rebuilds. You can further refine your index recovery using the KSORTSHARE option. You use the MAXLSORT option to specify concurrent log sorts and parallel MERGE phases. NGT Recover provides the ANALYZE ONLY option to view recovery resources and preview recovery plans without actually executing a recovery job. NGT Recover also provides a LOGSCAN command to gather information about the recovery and provide information that you can use to choose sort parameters.

NGT Recover uses the BMC BMCSORT technology for sorts. BMCSORT provides NGT Recover with more control of the sort process than external sort routines provide. This added control helps prevent memory-related problems during the sort process. NGT Recover allocates the amount of resources to each sort process based on the amount of work that NGT Recover determines that the sort process will perform. NGT Recover also dynamically detects excess available memory and allocates a percentage of it to the sort processes.

The recovery simulation feature, which requires a BMC Recovery Management for DB2 solution password, provides a non-destructive mechanism to exercise all recovery resources, including image copies, log records, and change accumulation files. This feature allows customers to ensure that all recovery resources are available to recover a table space or index without actually recovering it.

NGT Recover controls spaces by altering the DB2 status of the objects as necessary. Synchronization with BMC utility operations is provided through DB2 tables that are delivered and installed as part of NGT Recover.

NGT Recover does not use the DB2 subsystem buffers; however, it uses the DB2 subsystem to find information for the recovery process.
Using IBM DB2 resources

NGT Recover is fully compatible with IBM utilities and can rely on the same recovery resources and information during a recovery.

The DB2 resources that are used during a typical NGT Recover execution include the following items:

- Full image copy
- Incremental copies
- Active and archive logs
- Log ranges that are recorded in the SYSIBM.SYSLGRNX table
- Information that is stored in the DB2 catalog

**Note**

NGT Recover does not support DB2 concurrent copies.

NGT Recover and NGT Copy use an additional repository of information about recovery resources that are not used by the DB2 RECOVER utility. These additional resources include the following items:

- Image copies of a COPY NO index
- Data set level image copies of nonpartitioned COPY YES indexes
- Incremental index copies
- Instant Snapshot image copies
- Encrypted copies
- Online Consistent Copies (BMC Recovery Management for DB2 solution)
- Cabinet copies (BMC Recovery Management for DB2 solution)
- Migration file

Using NGT Recover with RECOVERY MANAGER

You can use the BMC RECOVERY MANAGER for DB2 product with NGT Recover. RECOVERY MANAGER provides an automated way to generate complex JCL for optimal recovery jobs and includes a facility to optimize multiple recovery jobs.

RECOVERY MANAGER provides multiple ways to designate groups of objects, including defined change accumulation groups and OBJECTSETs.

RECOVERY MANAGER creates groups using OBJECTSET. For information about the OBJECTSET syntax, see the OBJECTSET descriptions in “NGT Recover syntax” on page 83.
For complete details about the RECOVERY MANAGER product, see the
RECOVERY MANAGER for DB2 User Guide.

Using NGT Recover with R+/CHANGE ACCUM

The BMC R+/CHANGE ACCUM for DB2 product, with its ACCUM command,
offers the ability to create new recovery resources called change accumulation files.
When used during an NGT Recover execution, change accumulation files
significantly streamline normal and disaster recovery processes. R+/CHANGE
ACCUM enhances the speed at which recovery processes take place by extracting in
advance the log record data from the DB2 log that is required for recovery. Indexes
supported with NGT Copy copies and COPY YES indexes may also have log
changes accumulated.

When R+/CHANGE ACCUM is available, NGT Recover can process ACCUM and
RECOVER command statements in the same step for maximum efficiency in log
processing.

R+/CHANGE ACCUM is automatically installed with NGT Recover but requires its
own password for use or a Recovery Management solution password. (R+/CHANGE
ACCUM is part of the Recovery Management solution.)

For more information about R+/CHANGE ACCUM, see the R+/CHANGE ACCUM
for DB2 User Guide.

NGT Recover limitations

NGT Recover does not support the following items:

- IBM DB2 Version 9 or earlier
- IBM DB2 Version 10 running in compatibility mode or enabling new-function
  mode
- Recovery of catalog or directory spaces
- Recovery by using the ERROR RANGE and PAGE options (however, NGT
  Recover removes an existing error range after a successful table space or index
  recovery)
- Setting the check pending flag on a space recovered to a point in time prior to the
  definition of a column constraint
- Recovery of the DB2 objects that are used by NGT Recover:
— xxxx:x.CMN_BMCUTIL
— xxxx:x.CMN_BMCSYNC
— xxxx:x.CMN_BMCXCOPY
— xxxx:x.CMN_BMCHIST

xxxx is the creator name or ID.

- Index logical partition recovery
- Setting check pending flags for related spaces that are not recovered
- Use of an inline copy that has been merged with subsequent incremental copies
  The product tries to detect this condition by examining rows in SYSIBM.SYSCOPY indicating that an inline image copy is registered at the same START_RBA as a previous incremental copy. However, if those entries have been removed by the DB2 MODIFY utility or the NGT Copy MODIFY command, other errors may occur.
- REBUILD INDEX command if all of the following conditions are true:
  — A table is altered to add a column with the ROWID attribute.
  — An index is created with this new column.
  — A REORG is not performed.
- Use of the second copy of archive logs, as directed by the DSNZPARM ARC2FRST parameter
  Note
  The same facility is provided with the RESOURCE SELECTION LOGS option of NGT Recover.

- Tracker sites
- LISTDEF processing
- Inline statistics for REBUILD
- Same order of use for volumes from multivolume STOGROUP-defined spaces as DB2 RECOVER
  NGT Recover might use volumes from multivolume STOGROUP-defined spaces in a different order than DB2 RECOVER. You can control the order with the STOGROUP...USEORDER option.

Consider the following restrictions when working with NGT Recover and DB2:

- NGT Recover calls DSNUTILB for the following DB2 features:
— Indexes defined using an expression  
Some rebuild requests for indexes defined using an expression are passed to DSNUTILB. Some are processed natively, including, SUBSTR, UPPER, LOWER, DAY, MONTH, and YEAR.

However, recover requests are handled by NGT Recover.

— Indexes with RANDOM-order columns  
Rebuild requests for indexes with RANDOM-order columns are passed to DSNUTILB, but recover requests are handled by NGT Recover.

■ NGT Recover versions earlier than 10.1.00 will not work with DB2 Version 10.

■ NGT Recover versions earlier than 11.1.00 will not work with DB2 Version 11.

■ NGT Recover versions earlier than 12.1.00 will not work with DB2 Version 12.

■ If you recover an object to a point in time using DSNUTILB, you need to use DSNUTILB for subsequent recoveries of this object. You can recover this object using NGT Recover after you perform one of the following actions:
  — Make an image copy
  — Use REORG LOG YES
  — Use LOAD REPLACE LOG YES

**Note**

Do not use NGT Recover to back out over a range that includes an IBM backout recovery. Do not use IBM recover to recover over a range that includes an NGT Recover backout.

---

**Planning recovery**

NGT Recover is a powerful tool that aids in the speedy recovery of DB2 objects, but you are still responsible for recovery planning. You should become familiar with features and options that are offered by the utility and plan recovery strategies for different situations.

The features of NGT Recover have been designed with many different types of recovery situations in mind. Review all of the pertinent documentation and plan for media failures, application backouts, software failures, and other recovery scenarios that may occur.

When you are planning for recovery, consider using RECOVERY MANAGER for DB2 to generate JCL in advance, manage emergencies, and plan for disaster recovery.
The BMC Recovery Management for DB2 solution integrates the features of NGT Recover and RECOVERY MANAGER as well as other backup and recovery products for DB2 described in “Solution integration” on page 32. With this solution, you can benefit from all features of these individual products and additional features that are available when one Recovery Management component can rely on the presence of all others.

Recovery strategies

With NGT Recover you can control recovery with options that cover a wide range of recovery scenarios. These recovery scenarios are implemented by combining one or more NGT Recover commands and their options.

Note

All references to the RECOVER INDEX command or index recovery refer to the use of image copies and log. The REBUILD INDEX command or index rebuild refers to extracting and sorting index keys and building indexes. For more information, see the description of the INDEXLOG option (“INDEXLOG=NO” on page 584).

Using LOGSCAN for estimates

To help in planning for recovery, NGT Recover provides the LOGSCAN command. LOGSCAN gathers vital information about the log records that you can use to determine a recovery strategy. LOGSCAN uses only the LOG INPUT phase and writes the output to AFRPRINT or to a specified ddname (SYSSCAN). NGT Recover performs no log sort when you use LOGSCAN.

Note

The LOGSCAN command does not consider change accumulation files.

Output from the LOGSCAN command consists of

- Type of the object
- Name of the object
- Partition number (0, if nonpartitioned)
- Number of log records
- Byte count
- Calculated NUMREC EST value and AVGRECSZ
The LOGSCAN output does not include statistics for index rebuilds.

For more information, see “LOGSCAN” on page 253 and “Example 13: Using the LOGSCAN command” on page 423.

Using BACKOUT for point-in-time recovery without image copies

When you use NGT Recover to do a point-in-time recovery with the TOLOGPOINT option and the spaces are undamaged and current, you can use the BACKOUT option to employ a technique that uses the spaces and log records to restore spaces to a prior state.

Only the log between the point in time specified by TOLOGPOINT and the current log point is read and processed with this technique, which may substantially reduce log processing. In addition, the spaces are read for log backout processing, and image copy processing is completely eliminated. If indexes are not rebuilt and output image copies are not requested, only the pages updated by log records are read and written, which can greatly reduce I/Os and speed up the recovery. (For more information, see “Using copies and logs for index recovery” on page 48.)

To convert an existing partial recovery job to this technique, simply add the BACKOUT YES option to the usual syntax. Add the INDEXLOG YES or INDEXLOG AUTO option and NGT Recover assumes that the related indexes should be backed out to the same point in time.

Example

```
OPTIONS BACKOUT YES INDEXLOG YES  
RECOVER TABLESPACE PAYROLL.EMPLOYEE  
     TABLESPACE PAYROLL.RULES  
     TOLOGPOINT LASTQUIESCE  
RECOVER INDEX (ALL) TABLESPACE PAYROLL.EMPLOYEE  
RECOVER INDEX (ALL) TABLESPACE PAYROLL.RULES
```

When you select this strategy for potential point-in-time recovery, image copies may be avoided prior to batch runs without risking lengthy recovery times. Image copies do not need to be made of indexes to use the BACKOUT technique. (Indexes would still need to be rebuilt for media failure or disaster recovery.) COPY NO indexes are eligible for BACKOUT processing.

Note

A recovery that uses the BACKOUT option uses only the regular DB2 logs. A BACKOUT recovery does not use logs from the change accumulation files.
After a backout recovery, you must make a copy of the index before a recovery to current (by using a copy and log records) can be done. To maintain the ability to recover from log in case of a future media error, you can start an NGT Copy SHRLEVEL CHANGE copy running in parallel with other applications, as soon as the backout recovery is complete.

You can use the OUTCOPY YES option in your BACKOUT run. However, this request may degrade performance, because making an output copy causes all pages of the index space to be read.

**Note**

If you forego making a copy, you assume the risk that you will have to rebuild the index if it becomes damaged before your next image copy.

For more information, see “Using BACKOUT” on page 466.

**Comparing BACKOUT performance**

The backout recovery strategy is dramatically faster than most traditional forward recoveries as shown in the following figure. A backout recovery does not require image copies to perform a point-in-time recovery. Instead, it backs out the log records to undo or redo the changes that occurred between the selected point in time and the current point. This method returns the spaces and indexes to the required state without the overhead of restoring image copies, or rebuilding or restoring indexes.
Using BACKOUT AUTO recovery

When you request a point-in-time recovery, using the BACKOUT strategy is desirable. However, there are some restrictions that prevent an object from being backed out. The BACKOUT to forward strategy performs a BACKOUT on all eligible objects and automatically schedules a forward recovery for the ineligible objects, as well as any other objects that failed in the BACKOUT recovery step.

The BACKOUT installation option ("BACKOUT=AUTO" on page 580) defaults to AUTO. With this installation option set to its default value, if you are using a Recovery Management password, point-in-time recoveries automatically use the BACKOUT AUTO recovery strategy. If you are not using a Recovery Management password, point-in-time recoveries use a standard forward recovery. You can also request the BACKOUT to forward recovery strategy using the BACKOUT AUTO option ("BACKOUT" on page 122) on the OPTIONS command in the JCL.
Note

BACKOUT to forward recovery is a feature of the Recovery Management solution and the BACKOUT AUTO option requires the use of a valid Recovery Management solution password. For more information about BACKOUT AUTO recovery and Recovery Management, see the *Recovery Management for DB2 User Guide*.

Using copies and logs for index recovery

NGT Recover supports index recovery from full image copies, incremental image copies of indexes made by NGT Copy, logs, and change accumulation files. For traditional forward recoveries, you can make the required full image copies with NGT Copy or the IBM COPY utility. You can make incremental image copies of indexes with NGT Copy. NGT Recover can also use a copy that is produced with DSN1COPY when you specify the INCOPY...LOGPOINT keywords.

Point-in-time index recovery that uses logs as described above is also supported, allowing applications to be reset to a prior point in time without sorts of index keys. To use this feature, you must use the INDEXLOG YES or INDEXLOG AUTO option (as an installation option or OPTIONS command option) or the RECOVER INDEXSPACE command.

Performing forward recovery without an image copy

NGT Recover allows forward recovery from the log without using a copy to restore the space. You accomplish this type of recovery by using the RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX command (with INDEXLOG set to YES or AUTO in the installation options or on the OPTIONS command) and the LOGAPPLY ONLY option or the LOGONLY option. This feature allows you to restore a table space or index from a non-DB2 backup. The LOGAPPLY ONLY option also allows you to make forward adjustments to a table space or index after a point-in-time recovery.

Recovering with Instant Snapshot copies

NGT Recover supports forward recovery by using Instant Snapshot copies that are made by the BMC Next Generation Technology Copy for DB2 for z/OS product with the BMC EXTENDED BUFFER MANAGER (XB M) product. Instant Snapshot copies are data set level copies of table spaces or indexes that are made by using intelligent storage devices. These nontraditional copies are registered in SYSIBM.SYSCOPY as IBM FlashCopy (FC) copies if they are LP copies and in the BMCXCOPY table otherwise. Only full copies are allowed. To recover these copies, specify XBMID in
the installation options or on the OPTIONS command. Additionally, the SNAPSHOT
option adds support for Instant Snapshots with the INCOPY FULL syntax of the
RECOVER command.

For more information, see “Using Instant Snapshots” on page 469.

Using parallel MERGE phases and log sorts

NGT Recover allows multiple log sorts and parallel MERGE phases to run
concurrently in subtasks, which can provide an improvement in recovery
performance.

For each table space, the log sorts are distributed over the number of sorts specified
by the MAXLSORT option (“MAXLSORT integer” on page 136 and
“MAXLSORT=0” on page 588). MAXLSORT also determines the number of MERGE
phases that can run in parallel. When you combine index key sorts (see “Using
multitasking index key sorts and index rebuilds” on page 49) with log sorts, the
KSORTSHARE option (“KSORTSHARE” on page 137 and “KSORTSHARE=YES”
on page 586) determines whether NGT Recover shares the key sorts among table
space recoveries (MERGE phases) running in parallel.

For more information, see “Setting the MAXLSORT and KSORTSHARE options” on
page 481.

Using multitasking index key sorts and index rebuilds

NGT Recover allows multiple key sorts and multiple index rebuilds to run
concurrently in subtasks, which can provide a significant recovery performance
improvement.

For each table space, index keys for all indexes being rebuilt are distributed over the
number of sorts specified by the MAXKSORT option (“MAXKSORT integer” on
page 134 and “MAXKSORT” on page 587). When all keys are input to the sort for a
group of indexes, the build for each index is performed in a subtask. One subtask is
used for each group of indexes that were previously sorted. The KSORTSHARE
option, mentioned in “Using parallel MERGE phases and log sorts” on page 49 can
also improve your index recovery.

For more information, see “Setting the MAXKSORT option” on page 479.
Eliminating interim data sets for index keys

By using the NOWORKDDN option when you are rebuilding indexes, you can eliminate writing index keys to an intermediate data set before sorting. Use of the data set can also be eliminated if you do not code the WORKDDN option and you do not include a SYSUT1 DD statement in your JCL. Eliminating the interim data set allows NGT Recover to have BMCSORT run as a background task while the keys are extracted. Although this choice limits restart options, it improves the normal recovery time when rebuilding indexes.

Rebuilding nonpartitioned indexes on a partitioned table space

Another powerful execution option of the NGT Recover utility is the RECOVER UNLOADKEYS command, which constructs data sets that are used by the RECOVER BUILDINDEX command to rebuild nonpartitioned indexes on a partitioned table space. The RECOVER UNLOADKEYS command permits extraction of index keys from partitions in separate, concurrent jobs. You can set up multiple jobs to do the processing on the individual parts followed by a job to rebuild the nonpartitioned index. This approach runs in less time than a single job that reads all of the partitions and rebuilds the index.

Each unloaded keys data set is sorted individually in RECOVER UNLOADKEYS. These sorted files are merged by RECOVER BUILDINDEX, which results in multiple smaller sorts that could improve performance and reliability in some cases. This method may be particularly useful when you are working with very large table spaces.

Using resource selection

NGT Recover resource selection features provide a way for you to control how NGT Recover accesses and orders the multiple images of the DB2 log, image copies, and BMC change accumulation files during recovery. These powerful features provide additional control, which you can use to avoid resources that are unavailable, avoid contention, and use copies registered for one site type while running as another site type.

For example, to avoid a single point of failure, DB2 subsystems are generally run with dual active logs. Dual archiving is available as well. In most cases, NGT Recover attempts to use Active 1 and then Active 2, followed by Archive 1 and Archive 2. The first available file that contains the log records required is actually read. You can use the NGT Recover resource selection features to control whether, and in what order, to use each source of log records.
You can create multiple copies of image copies and change accumulation files. You can create two pairs, one designated LOCAL and one REMOTE. These copies are classified as the following types:

- LOCAL PRIMARY
- LOCAL BACKUP
- REMOTE PRIMARY
- REMOTE BACKUP
- SYSTEM BACKUP
- FlashCopy (image copies only)

Normally, NGT Recover uses only LOCAL copies if the DB2 subsystem is running as a LOCAL site and only REMOTE copies for the REMOTE site. PRIMARY is normally considered before BACKUP unless the TOCOPY option indicates a backup. You can use the NGT Recover RESOURCE SELECTION option to control which copies to attempt to use, and in what order, regardless of whether they are registered as LOCAL or REMOTE.

**Previewing recovery activity**

You can use the NGT Recover ANALYZE ONLY option to preview recovery activity before executing a recovery. ANALYZE ONLY reports information about image copy data sets, log data sets, change accumulation files, and log ranges that NGT Recover will use for recovery. NGT Recover includes an optional tape pick list and recall list in its reporting.

Information about the record sizes for work files and index sorts is also provided. You can use these statistics to accurately allocate space for log sort and index sort work files and index key work data sets. As an option, you can recall archived resources during an ANALYZE ONLY run.

This feature provides some predictability of recovery operations by providing to you the means to examine the recovery procedures and strategies that you wish to implement.

The LOGSCAN command provides information about the number and size of log records required for recovery. You can use this information to size the log sort. A printed report describes the log records for each object that is named in the LOGSCAN command.
Testing recovery with simulation

You can use the SIMULATE YES option on the OPTIONS command to go a step beyond previewing recovery activity to testing it.

This non-destructive option (“SIMULATE YES” on page 132) performs most of the work of a normal recovery. After reading and merging image copies, log files, and change accumulation files, NGT Recover discards the output and leaves the underlying table space unaffected. Using this option, you can see exactly how a recovery will run without sacrificing data availability.

**Note**

Simulation is a feature of the BMC Recovery Management for DB2 solution and requires a valid solution password.

Recovery simulation provides a way for you to validate that you can recover your application data. You can use recovery simulation to verify that needed recovery resources are valid and available and that log apply can be performed.

The following synonyms are required in NGT Recover syntax when you specify OPTIONS SIMULATE YES:

- SIMRCVR TABLESPACE, INDEXSPACE, or INDEX for RECOVER TABLESPACE, INDEXSPACE, or INDEX (“SIMRCVR” on page 167)
- SIMRBLD INDEX for REBUILD INDEX (“SIMRBLD” on page 219)
- SIMRCVR UNLOADKEYS for RECOVER UNLOADKEYS (“RECOVER UNLOADKEYS” on page 237)
- SIMRCVR BUILDINDEX for RECOVER BUILDINDEX (“RECOVER BUILDINDEX” on page 246)

You can use RECOVERY MANAGER for DB2 to build the JCL for recovery simulation. For more information, see the RECOVERY MANAGER for DB2 User Guide.

Operational considerations

When you specify OPTION SIMULATE YES in an NGT Recover job, NGT Recover proceeds with all recovery activities that are not destructive. Destructive actions that could affect real database objects or the recovery environment are simulated, suppressed, or executed in the temporary data sets. The following activities are included:

- Reading and sorting log records
- Merging image copies, log records, and change accumulation files
Extracting index keys

NGT Recover does not stop any table spaces or indexes during simulation mode.

In the following recovery scenarios, NGT Recover creates and uses temporary data sets (SIMDBC or SIMDBD), instead of real spaces (DSNDBC or DSNDBD), for simulation:

- Recoveries from Inline, instant snapshot, or flashcopy image copies
- Backout, LOGONLY, or LOGAPPLY ONLY recoveries

When you simulate recovery of inline, instant snapshot, or flashcopy image copies, NGT Recover snaps the image copy into temporary data sets to process the simulation.

When you simulate a backout, LOGONLY, or LOGAPPLY ONLY recovery, NGT Recover snaps the original spaces into temporary data sets to process the simulation.

NGT Recover performs the following tasks for table spaces and for indexes:

- Reads full and incremental image copies that are required for recovery
- Reads change accumulation files that are required for recovery
- Simulates applying log records, change accumulation log records, or both
- Extracts index keys for any index rebuilds that are required

NGT Recover processes the extracted, sorted keys and builds each index page in storage.

NGT Recover does not perform the following activities:

- Write any output pages to DASD (if you are not using temporary data sets for a simulation of recovery)
- Write any output image copies
- Write to BMCSYNC or BMCUTIL
- Register any events in SYSIBM.SYSCOPY
- Set or reset any pending flags

**Limitations**

Use of recovery simulation has the following limitations, most of which result from not performing any disruptive activities on the spaces that are involved.

- The following NGT Recover features are not supported in recovery simulation:
  - ANALYZE ONLY or ANALYZE SCANONLY
  - LOGSCAN
— Point-in-time recovery

**Note**
Simulation of point-in-time recovery is indirectly supported by specifying a log point with the SIMULATE option. In this case, NGT Recover simulates recovery of all objects in the SYSIN to the same log point. Simulation of point-in-time recovery is also provided by the OPTION RECOVERYPOINT option if the BMC Recovery Management for DB2 solution is installed.

- Simulation runs are not restartable.
- The requested recovery scenario can require unloading keys from the real table space (partition) if REBUILD or UNLOADKEYS is requested for the table space (partition) not recovered in the same step. Such unload operations involving real table spaces (partitions) will be suppressed in simulation mode.
- Objects may be in any initial status and there is no impact on the status or contents of the real database objects.

**Recovering while activity continues**

Activity can continue on the original table space while a recovery is being performed to a consistent or arbitrary point on a copy of the table space. You perform this task by naming new data sets for the recovery of the DB2 table space that is subject to errors.

After recovery is complete, apply the log to make the new table space current. Renaming the new data sets will cause transactions to access the new table space.

**BMCXCOPY registrations**

Full image copies of indexes are registered in the BMCXCOPY table if one of the following situations applies:

- The index is defined with COPY NO.
- The index is a nonpartitioning index with data set level (DSNUM n) copies, even if the nonpartitioning index is defined as COPY YES.

These registrations are made in BMCXCOPY rather than in SYSCOPY because DB2 does not allow registration of such image copies in SYSCOPY.

Incremental index copies are always registered in BMCXCOPY.
Instant Snapshot copies of table spaces and indexes, which are made by BMC Next Generation Technology Copy for DB2 for z/OS with EXTENDED BUFFER MANAGER for DB2, are registered in the BMCXCOPY table with an STYPE of V. When a local primary Instant Snapshot copy is registered in BMCXCOPY and a local backup copy is made, the local backup copy is also registered in BMCXCOPY, even if it is a regular DB2 image copy.

Encrypted image copies, which are made by NGT Copy, are registered in the BMCXCOPY table with an STYPE of e. For more information about encrypted copies and their registration, see “Recovering encrypted copies” on page 506.

When ICTYPE=P, BACKOUT recovery is registered in the BMCXCOPY table with an STYPE of B.

Cabinet copies, which are made by the Recovery Management solution, are registered in BMCXCOPY with a COPY_TYPE value of C. For more information about cabinet copies, see “Using cabinet copies” on page 60.

Migration files which are created by the NGT Copy EXPORT command are registered in BMCXCOPY.

Simulated recoveries and rebuilds, and INDEP OUTSPACE recoveries and rebuilds, are registered in BMCXCOPY with the following ICTYPE / TTYPE:

<table>
<thead>
<tr>
<th>Action</th>
<th>ICTYPE</th>
<th>TTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER to PIT with INDEP OUTSPACE</td>
<td>p</td>
<td>I</td>
</tr>
<tr>
<td>SIMRCVR to PIT</td>
<td>p</td>
<td>S</td>
</tr>
<tr>
<td>RECOVER to CURRENT with INDEP OUTSPACE</td>
<td>e</td>
<td>I</td>
</tr>
<tr>
<td>SIMRCVR to CURRENT</td>
<td>e</td>
<td>S</td>
</tr>
<tr>
<td>REBUILD INDEX with INDEP OUTSPACE</td>
<td>b</td>
<td>I</td>
</tr>
<tr>
<td>SIMRBLD</td>
<td>b</td>
<td>S</td>
</tr>
</tbody>
</table>

The BMCXCOPY table is described in “BMCXCOPY table” on page 626.

Creating copies from recovery resources

You can use NGT Recover to create and register table space or index image copies based on other recovery resources without accessing or affecting the availability of the object. For example, you can create image copies from other copies, logs, and change accumulation files.
This type of copy can be created as of a prior point in time or at the current time. By creating copies from other recovery resources, you can create copies registered at the point in time that an application started while the application is running. If you must recover to that point, you will have a registered copy available. If you need a copy at a given point in time, you can create and register the copy as needed.

**Overriding registered copies**

If you have valid copies that are not registered in SYSIBM.SYSCOPY or the BMCXCOPY table, you can use NGT Recover to override the copies that are specified in these tables with a list of one or more copies to be used instead.

**Recovering LOB spaces with NGT Recover**

NGT Recover recovers or rebuilds all components of LOB table spaces and indexes. The BACKOUT function is not available for LOB auxiliary table spaces.

**Recovery of LOG NO LOB auxiliary table spaces**

When NGT Recover recovers a LOB defined as LOG NO, NGT Recover leaves the space in AUXW status if it detects that there were any unlogged updates in the range of the recovery.

Any LOB objects that were updated with unlogged updates are invalidated and reported by the message BMC96124W:

```
BMC96124W LOG NO UPDATE FOR ROWID(X'50633BFCD66BD08B210401581C30404040').
VERSION 2
```

DB2 allows SQL access to all other rows in the table even though the space is in AUXW status. DB2 also allows SQL access to all other columns of the affected row other than the invalidated LOB. Also, DB2 allows SQL to update the LOB value itself to a new value.

The format of the BMC96124 message allows for an easy cut and paste to SQL using the ROWID value as a predicate as shown in the following statement:

```
SELECT EMPNO,FNAME,LNAME FROM WRPDB83.WRPTB83A
WHERE ROWID1 = ROWID(X'50633BFCD66BD08B210401581C30404040')
EMPNO FNAME LNAME
------- -------- --------
 1   JOHN   SMITH
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```
**Point-in-time recoveries**

You should always recover a LOB base table space and its related auxiliary table space or spaces and indexes together and to a consistent point in time. Recovery of any individual component to any other point in time might leave the space in an error condition that is difficult to resolve.

If the space is left in any of the exception statuses described in the following list, use the DB2 CHECK DATA utility to remove the error status. Consult the DB2 documentation for instructions for using this utility.

- If NGT Recover recovers the base table space to a previous point in time and does not recover the auxiliary table space to the same point in time, NGT Recover leaves the base table space in ACHKP status.

- If NGT Recover recovers the auxiliary table space to a previous point in time and does not recover the base table space to the same point in time, NGT Recover leaves the base table space in ACHKP status and the auxiliary table space in CHKP status.

- If NGT Recover recovers the base and auxiliary table spaces to the same point in time, but that point in time is not a consistent point (such as a quiesce point), NGT Recover leaves the base table space in ACHKP.

- If NGT Recover recovers the base and auxiliary table spaces to the same point in time, and if that point in time is a registered quiesce point, and if all related indexes are rebuilt or recovered to the same point in time, no error statuses are set.

- If NGT Recover recovers the base and auxiliary table spaces to the same point in time specified in OPTION RECOVERYPOINT syntax (available only with the BMC Recovery Management for DB2 solution), and if all related indexes are rebuilt or recovered to the same point in time, no error statuses are set.

**Automatic fallback without restarting**

During the recovery of a space, fallback is the process of using an equivalent or earlier image copy to replace an invalid or unavailable image copy. Automatic fallback extends the NGT Recover fallback capabilities of an initial execution to all situations involving standard image copies or Instant Snapshot copies, except for recoveries that use the INCOPY option or the RECOVER UNLOADKEYS command. The automatic fallback feature of NGT Recover can save you time and increases the usability of the utility.
Note

Automatic fallback does not occur if you specify:
- INCOPY option on a RECOVER command
- RECOVER UNLOADKEYS command
- REBUILD INDEX command with the WORKDDN option

For more information, see “Fallback recovery with NGT Recover” on page 465.

Using incremental index image copies

Using incremental copies of indexes made by NGT Copy in recovery has the following benefits:

- Reduces the need for excessive DB2 log application for an index space recovery
- Reduces the amount of data that must be copied to disk or tape in order to maintain current copies
- Reduces the volume of image copy data to manage
- Improves recovery times compared to rebuilding indexes or to using only full index image copies

Note

To use the incremental index copies made with NGT Copy, the same version of NGT Recover is required. If you attempt a recovery using incremental index copies and earlier versions of NGT Recover, unpredictable results can occur.

NGT Recover recognizes incremental copies of indexes recorded in BMCXCOPY, includes them in NGT Recover analysis and planning, shows them in the NGT Recover object summary, and uses them whenever appropriate to speed up recovery of indexes.

When you use incremental index copies for recovery, note the following considerations:

- Incremental index space copies that are produced by NGT Copy are registered in the BMCXCOPY table.
- Because index spaces lack the ability to identify modified pages, incremental index space copies are always produced using the READTYPE FULLSCAN method.
Because index spaces lack the ability to identify modified pages, the RESETMOD option for copies does not apply. All index copies are RESETMOD NO.

NGT Recover processing for cumulative incremental image copies applies to incremental index copies. For more information, see “Recovery using cumulative incremental image copies” on page 78.

Restrictions on incremental index copies

The following restrictions apply to incremental index copies:

- You cannot use DSNUM n incremental copies of nonpartitioned indexes in a DSNUM ALL recovery.
- NGT Copy and NGT Recover do not support incremental index copies of catalog and directory.

Incremental copies of nonpartitioned indexes and recovery

When you make incremental copies for nonpartitioned indexes, you should make these copies using the NGT Copy IXDSNUM=ALL option.

If you make the copies using the NGT Copy IXDSNUM=DATASET option, your NGT Recover statements must specify the data set number on the RECOVER statement.

If you copy a nonpartitioned index with IXDSNUM=ALL, examples of your NGT Recover syntax are:

RECOVER INDEX IX.A

or

RECOVER INDEX IX.A DSNUM ALL

If you copy a nonpartitioned index with IXDSNUM=DATASET, examples of your NGT Recover syntax are:

RECOVER INDEX IX.A DSNUM n

or

RECOVER INDEX IX.A DSNUM n:n (more than one data set to be recovered)

For more information about IXDSUM option, see the BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual.
Using cabinet copies

NGT Recover supports the cabinet copy feature of the BMC Recovery Management for DB2 solution.

**Note**
Because the cabinet copy feature is part of the BMC Recovery Management for DB2 solution, making cabinet copies and using them for recovery requires use of a valid Recovery Management solution password. For more information about cabinet copies and Recovery Management, see the Recovery Management for DB2 User Guide.

Cabinet copies provide a performance enhancement when you are copying a large number of spaces. In such cases, the overhead to open and close each copy data set can be a significant component of overall run time.

Cabinet copies allow you to copy all the spaces in a group into a single data set called a cabinet file. The cabinet file is allocated and deallocated only once, regardless of the number of objects that are copied to or recovered from the cabinet file. Because there is no file opening or closing for each space in the cabinet file, the file header and trailer records, including the EOF markers, are omitted from cabinet files, and performance is greatly improved. You can copy the cabinet files to either DASD or tape.

In addition to providing a performance enhancement, cabinet copies can save resources because using cabinet copies can

- Save disk space because of the efficient use of space within a cabinet file
- Reduce the number of MVS catalog entries

NGT Recover automatically detects and uses cabinet copies if they are available for recovery for any spaces in the cabinet copy. Phases to recover objects will be planned in the order that the object image copies appear in the cabinet copy unless this would conflict with other planning restrictions such as change accumulation files or output copies.

**Restrictions for cabinet copies**

The following restrictions apply to cabinet copies:

- A minimum z/OS level of 1.7 is required for cabinet copies made to disk.
- For a recovery using cabinet copies, you must use NGT Recover version 8.1.00 or later.
- An image copy contained within a cabinet copy must have a row present in BMCXCOPY on the subsystem on which the recovery is run.
The INCOPY option of NGT Recover does not support cabinet copies.

**Note**
If you have only a cabinet copy and need to do a drop recovery, you can use the NGT Copy COPY IMAGECOPY command to unstack a cabinet copy and register the copies in SYSCOPY. You can then use the individual copies with the INCOPY option.

Online consistent copies, another feature of the Recovery Management solution, do not support cabinet copies.

You cannot make cabinet copies using the Instant Snapshot functionality.

### Registration of cabinet copies

Because cabinet copies can be recovered only by using NGT Recover, they are registered in the BMCXCOPY table. A COPY_TYPE value of C indicates the copy is a cabinet copy. A row is created for each member within the cabinet copy and each member has the same DSNAME. The DSNAME is the name of the cataloged cabinet file as indicated in the NGT Copy OUTPUT command. The cabinet members are individually registered in BMCXCOPY as uncataloged. The cabinet file is always cataloged.

**Note**
Copies made using DSNUM ALL reduce the number of entries in BMCXCOPY for spaces with many partitions.

### Using Online Consistent Copy

Online consistent copies, copies made without any outage and that are physically and transactionally consistent, are a feature of the BMC Recovery Management for DB2 solution and are registered in the BMCXCOPY table. These copies, which are used for recovery to copy, are also useful for migration. Recovery Management introduces the use of online consistent copies and log apply for forward recovery.

**Note**
Because the Online Consistent Copy feature is part of the BMC Recovery Management for DB2 solution, making online consistent copies and using them for recovery requires use of a valid Recovery Management solution password. For more information about Online Consistent Copy and Recovery Management, see the *Recovery Management for DB2 User Guide*.

NGT Recover supports the use of online consistent copies and log apply for forward recovery to the current time or to a point in time, as well as recovery to a copy.
Note

Note the following considerations:

- NGT Recover supports online consistent copies with log apply when you do not specify INCOPY.

- NGT Recover supports online consistent copies with INCOPY TOCOPY specified.

- NGT Recover does not support online consistent copies when you specify both INCOPY and log apply (RBA or LOGPOINT specified in the INCOPY specification).

Extra processing is required to make online consistent copies and to recover using log applied to an online consistent copy, so you should make online consistent copies and use them for recovery only when doing so adds value to your process, such as when there is a requirement for migration or a recovery to a copy.

You can make online consistent copies on either snappable or nonsnappable storage. However, online consistent copies generally perform better when they are on snappable storage. If consistent copies are on nonsnappable storage, you need to specify the DATAMVR installation option (“DATAMVR=” on page 582) or specify the DATAMVR option on the OPTIONS statement (“DATAMVR programName” on page 129) to override the installation option. The DATAMVR option provides XBM with the name of the program to use to copy a data set.

If you request a RECOVER TOCOPY using an online consistent copy that is not available, the fallback recovery is a TIMESTAMP recovery.

Note

If you perform a RECOVER TOCOPY using an online consistent copy, you should make a new copy after the recovery completes. After the RECOVER TOCOPY, you will not be able to use copies made before the START_RBA of the online consistent copy.

Using a system-level volume backup

DB2 supports the BACKUP SYSTEM command, which backs up the entire DB2 system using volume snapshots. To recover an individual space from a system-level volume backup, you can use RECOVER in NGT Recover and specify SB (for system backup) in the RESOURCE SELECTION for image copies. Also, COPY IMAGECOPY in NGT Copy can make a standard copy of the space from a system backup.
For more information about using the COPY IMAGECOPY to create a standard copy from a system-level volume backup, see Chapter 2 of the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*.

**Migrating or importing data**

With the Copy Migration feature of the Recovery Management solution, you can use the NGT Recover MIGRATE and IMPORT commands, along with the migration file from the NGT Copy EXPORT command, for simplified data migrating or importing. The process includes the following benefits:

- Supports importing copies that use BMC proprietary formats
- Does not require that you manage data sets to keep up with current copies
- Does not require that you know the OBIDs from the source system
- Imports applications and object sets as a single unit
- Does not require the import of unchanged objects (to save time)
- Supports the use of older EXPORT files to back-date imports
- Provides a less error prone, and therefore, more accurate data migration (for example, less likely to use the wrong data set name or OBIDs)

**Note**

To receive all of the benefits provided above, you must have one of the following valid passwords:

- A Recovery Management solution password
- A Database Administration solution password

For more information, see the following sources:

- The *Recovery Management for DB2 User Guide*
- The EXPORT command description in the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*
- “MIGRATE” on page 270 and “IMPORT” on page 290

**Solving other problems with NGT Recover**

You can use NGT Recover to perform the following tasks:

- Recovery from a dropped table space
- Migration of data to another subsystem
- Creation of a shadow (parallel) space on the same subsystem
- OBID translation
- Specification of input copies
Override of log ranges
Conversion of database objects to new definitions with minimal outage

**Note**
Versioned table spaces present special problems during drop recovery or data migration. See “Handling DB2 versioning information” on page 513.
Operational considerations

This chapter discusses issues and concepts you should consider when using BMC Next Generation Technology Recover for DB2 for z/OS in an operational environment.

Operating environment

NGT Recover requires the operating environment described in this section.

DB2 support

This version of NGT Recover supports IBM DB2 Versions 10, 11, and 12. DB2 Version 10 is supported only in new-function mode.

System requirements

This version of NGT Recover supports all versions of the IBM z/OS system that IBM supports and that are compatible with IBM DB2 Versions 10, 11 and 12.

Following are some considerations based on operating system level:

- If you are running z/OS Version 1.7 or later, you can use NGT Recover to recover table spaces and index spaces by using large format (greater than 64 K tracks) archive logs and image copies.

- If you are running z/OS Version 1.10 or later, NGT Recover supports extended address volumes (EAVs) for DB2 spaces, Instant Snapshot copies, active logs, and the BSDS.

- If you are running z/OS Version 1.11 or later, NGT Recover supports archive logs and sequential image copy data sets in the cylinder-managed portion of EAVs.
Note
You cannot use an image copy made to the cylinder-managed portion of an extended address volume (EAV) under z/OS Version 1.11 on z/OS Version 1.10 because z/OS Version 1.10 does not support sequential data sets in the cylinder-managed portion of an EAV.

- IDCAMS and DSNUTILB must be available in the system LPA, LINKLIST, JOBLIB, or STEPLIB.

zIIPs

NGT Recover ascertains if the system has IBM z Integrated Information Processors (zIIPs). If it does, NGT Recover offloads as much processing as possible to zIIP processors.

Tip
When you use zIIPs, BMC recommends that you specify IIPHONORPRIORITY=YES in the IEAOPT member of SYS1.PARMLIB.

You can disable use of zIIP processors by specifying OPTIONS ZIIP DISABLED.

Setting the MEMLIMIT parameter

The following products and components require above-the-bar memory and might abend if sufficient memory is not available:

- ALTER
- BMCSORT
- CATALOG MANAGER
- CHANGE MANAGER
- CHECK PLUS
- NGT Copy
- DASD MANAGER PLUS
- High-speed Apply Engine
- LOADPLUS
- Log Master
- NGT Recover
- RECOVERY MANAGER
- REORG PLUS
- UNLOAD PLUS
In z/OS versions before 1.10, the default value for the System Management Facility (SMF) MEMLIMIT parameter is 0; a value of 0 means that no address space can use virtual storage above the bar. In z/OS Version 1.10 and later, the default value is 2 GB.

For most jobs, BMC recommends a value of at least 1 GB for the MEMLIMIT parameter. However, if you are operating on LOB or XML data, BMC recommends a value of at least 32 GB.

This value is set in member SMFPRMxx in SYS1.PARMLIB. Use any of the following methods if you need to override the default value:

- Specify the MEMLIMIT parameter in the JCL.
- Specify REGION=0M in the JCL.
- Use the SMF IEFUSI exit.

## Software requirements

This version of NGT Recover has the following requirements for additional IBM or BMC software:

- You must have a minimum of version 12.1.00 of the BMC DB2 Solution Common Code (SCC) installed.

- If you want to offload eligible processing to a zIIP, you must have installed a minimum of version 6.2 of either XBM or SUF.

  If you use the XBMID option to specify a particular XBM subsystem, that subsystem must be at this maintenance level. If you do not specify a particular XBM subsystem and ZIIP ENABLED is in effect, NGT Recover searches for an XBM subsystem at this level.

- To use any features that invoke DSNUTILB, you must have installed the IBM DB2 RECOVER utility.

  **Note**

  If NGT Recover invokes DSNUTILB, the call displays in the AFRPLAN and AFRPRINT output files, and DSNUTILB messages display in the AFRPRnnn output files.
Authorization

Using NGT Recover requires that you have authorization within DB2 and (in some cases) through your system security package (such as RACF) that is sufficient to access DB2 resources and perform the tasks accomplished during NGT Recover processing.

Authorization verification mechanisms

If the DB2 DSNX@XAC authorization exit is available for your system, NGT Recover uses this exit to verify authorization for external access. The exit is available from the following sources:

- IBM provides a sample exit with DB2 for the IBM Resource Access Control Facility (RACF) component.
- CA Technologies provides the DSNX@XAC exit with CA-ACF2 Security for DB2 and CA-Top Secret Security for DB2.

BMC recommends this mechanism for implementing external security. The access control authorization exit must be available in the STEPLIB, JOBLIB, linklist, or in the SYS3.DSN exit.

If the DSNX@XAC exit is not available, NGT Recover uses the standard DB2 method to check security.

DB2 authority

To run NGT Recover, you must have one of the following authorizations:

- Installation SYSADM or SYSADM authority
- Sufficient authority to run the NGT Recover application plan, and one of the following authorizations:
  - SYSCTRL or system DBADM
  - DBADM or DBCTRL authority for the databases containing the named table spaces
  - RECOVERDB, DISPLAYDB, STARTDB, and STOPDB authority for the databases containing the named table spaces and index spaces
— DATAACCESS, DISPLAYDB, STARTDB, and STOPDB authority for the databases containing the named tables spaces and index spaces

— IMAGCOPY and DISPLAYDB authority to execute OUTFCOPY only

Because NGT Recover uses the BMC dynamic bind technology, the OWNER of the plan must be authorized to EXECUTE each package in the plan at the time dynamic bind is performed. If you do not modify the OWNER of the NGT Recover plan specified during installation, you should not need to be concerned with this requirement.

**System authority**

Because NGT Recover does not run as part of the DB2 subsystem, you must have system authority similar to that of DB2 to use NGT Recover.

If the underlying data sets of a table space or index space are RACF or similarly protected, you must have sufficient authority to access and modify the data set. If a table or index space is STOGROUP-defined and the corresponding ICF catalog is RACF or similarly protected, you must also have sufficient authority to access and update the operating system catalog. The minimum levels of authority shown in Table 1 on page 69 are required when you use the following settings:

- The installation option OPNDB2ID is set to NO.
- OPNDB2ID is set to YES and a security system other than RACF is used.

**Table 1: System authority requirements**

<table>
<thead>
<tr>
<th>Table or index space definition</th>
<th>Minimum levels of authority required to access and update data sets</th>
<th>Minimum levels of authority required to access and update the operating system catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCAT</td>
<td>Control</td>
<td>None required</td>
</tr>
<tr>
<td>STOGROUP</td>
<td>ALTER or Control</td>
<td>UPDATE (if data set authority is ALTER) or ALTER (if data set authority is Control)</td>
</tr>
</tbody>
</table>

If active logs will be read and OPNDB2ID = NO, the ID running the job needs ALTER authority.

If OPNDB2ID is set to YES and RACF is used, these authorities are not required; in this case, the RACF ID for DB2 is used when opening the DB2 data sets or catalog. For more information about the OPNDB2ID installation option, see “OPNDB2ID=YES” on page 589.

If DB2 is specified in the RACF started procedures table (ICHRIN03) as a privileged or trusted task and no user ID is associated with the DB2 address space, you cannot
use OPNDB2ID to allow NGT Recover to access the DB2 data sets. In this case, the user running NGT Recover must have RACF authority to access the data sets needed for recovery.

The NGT Recover option OPNDB2ID works under data sharing only if all RACF IDs for the members of a group are the same. Authorizations for the bootstrap and log data sets must also be the same.

In addition to the traditional DB2 DSNDDBC and DSNDBD data sets, NGT Recover requires the following access:

- SIMDBC for simulated recovery
- SIMDBD for simulated rebuild
- BMCDB for INDEP OUTSPACE recovery
- BMCDBD for INDEP OUTSPACE rebuild
- REBUILD with SHRLEVEL CHANGE

If OPNDB2ID=YES, this access must be granted to DB2's RACF ID. If OPNDB2ID=NO, this access must be granted to the user running NGT Recover.

### APF authority

NGT Recover uses system services that require APF authorization. Accordingly, NGT Recover must reside in an APF-authorized library.

All load modules loaded by NGT Recover must be authorized and must reside in APF-authorized libraries, as follows:

- IDCAMS
- DSNUTILB
- Data Facility Product (DFP) module IGWASYS (which generally resides in SYS1.CSSLIB)
  - For DFP Version 3.2 or earlier, this module is called IGWAQSMS.

### Table space and index space status

If the output from a table space or index space recovery is to a DB2 data set, the initial status of the target space must be acceptable to NGT Recover before a recovery can start.

However, if SIMULATE YES is specified or INDEPENDENT OUTSPACE is specified to direct the recovery output of a table space to a non-DB2 data set, the initial status...
of the space is not a factor in the recovery and is ignored by NGT Recover. Also, no status changes are made to the space. But if you do not specify INDEPENDENT INTABLESPACE when you use the INDEPENDENT OUTSPACE option with REBUILD INDEX, the table space is stopped and started as READ ONLY (RO). In this case, the initial table space status is a factor.

If the recovery is to a DB2 data set and the initial status of the target table space or index space is not acceptable, NGT Recover terminates. If the status is acceptable, NGT Recover stops all of the spaces during the ANALYZE phase to ensure exclusive use of the spaces during the recovery.

NGT Recover sets the appropriate pending status (RECP or RBPD) for each space in anticipation of processing the space. When the recovery of the space is completed successfully, the pending status is turned off. If the job fails, a space may be left in a pending status even though it was never processed.

If you set the RDB2STAT parameter to NO, NGT Recover will leave the spaces stopped to ensure that you take some action before allowing the spaces to be used. For example, you may want to ensure that all related spaces have been recovered to the same point, or you may want to take image copies for future recovery if copies were not taken during processing.

If NGT Recover determines that the spaces can be used when a recovery is complete, the initial status is restored if the RDB2STAT parameter is set to YES. (You can also use RDB2STAT(RW) or RDB2STAT(RO) on the EXEC statement as described on “RDB2STAT override parameter (rdb2Stat)” on page 317 to set the space in RW or RO status at the end of the run.)

If a table space status is copy pending before a recovery and an output image copy is successfully registered, NGT Recover removes the COPY status. NGT Recover does not remove the COPY status for an image copy that was created with the OUTCOPY ONLY option.

If REBUILD INDEX or RECOVER BUILINDEX of a nonpartitioned index space on a partitioned space fails during the BUILD phase, NGT Recover sets page set recover pending (PSRCP) or page set rebuild pending (PSRBD) status on the index space.

**Initial table space and index space status**

The following table shows the action taken by NGT Recover for acceptable and unacceptable initial DB2 table space status.
<table>
<thead>
<tr>
<th>NGT Recover command</th>
<th>NGT Recover action for table spaces with acceptable initial status</th>
<th>NGT Recover action for table spaces with unacceptable initial status</th>
<th>Unacceptable initial table space status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER TABLESPACE (with or without index recovery or unload keys) without BACKOUT option</td>
<td>Issues the DB2 command STOP AT (COMMIT) for the table space</td>
<td>NGT Recover terminates</td>
<td>RESTART UTUT UTRW UTRO AREST RESP REFP b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECOVER TABLESPACE (with or without index recovery or unload keys) with BACKOUT option</td>
<td>Issues the DB2 command STOP AT (COMMIT) for the table space</td>
<td>NGT Recover terminates</td>
<td>RECP LPL GRECP DEFER WEPR REFP UTUT UTRW UTRO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REBUILD INDEX SHRLEVEL REFERENCE (but not REBUILD INDEX SHRLEVEL CHANGE)</td>
<td>Issues the DB2 command STOP AT (COMMIT) for the table space and restarts in ACCESS(RO)</td>
<td>NGT Recover terminates</td>
<td>RESTART UTUT UTRW RECP DEFER LPL GRECP AREST RESTP WEPR REFP</td>
</tr>
</tbody>
</table>

a When the recovery is to a non-DB2 data set, the initial status of a table space is ignored and no status changes are made.

b Acceptable if the recovery is to a point in time.

Table 3 on page 73 shows the action taken by NGT Recover for acceptable and unacceptable initial DB2 index space status.
Table 3: NGT Recover actions for access to DB2 index spaces

<table>
<thead>
<tr>
<th>NGT Recover command</th>
<th>NGT Recover action for index spaces with acceptable initial status</th>
<th>NGT Recover action for index spaces with unacceptable initial status</th>
<th>Unacceptable initial index space status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER INDEX or RECOVER INDEXSPACE without BACKOUT option</td>
<td>Issues the DB2 command STOP AT (COMMIT) for the index space</td>
<td>NGT Recover terminates</td>
<td>RESTART UTUT UTRW UTRO AREST RESTP REFP b</td>
</tr>
<tr>
<td>RECOVER INDEX or RECOVER INDEXSPACE with BACKOUT option</td>
<td>Issues the DB2 command STOP AT (COMMIT) for the index space</td>
<td>NGT Recover terminates</td>
<td>RECP RESTART UTUT UTRW UTRO RBDP PSRCP PSRBD WEPR REFP</td>
</tr>
<tr>
<td>REBUILD INDEX SHRLEVEL REFERENCE or RECOVER BUILDINDEX</td>
<td>Issues the DB2 command STOP AT (COMMIT) for the index space</td>
<td>NGT Recover terminates</td>
<td>RESTART UTUT UTRW UTRO AREST RESTP</td>
</tr>
<tr>
<td>RECOVER UNLOADKEYS</td>
<td>No action is required, any status is acceptable</td>
<td>No action is required, any status is acceptable</td>
<td>N/A</td>
</tr>
</tbody>
</table>

a When the recovery is to a non-DB2 data set, the initial status of a table space is ignored and no status changes are made.

b Acceptable if the recovery is to a point in time.

When the initial status is acceptable, NGT Recover issues DB2 commands that give control of the space to NGT Recover and allow the recovery to start. When the initial status is not acceptable, NGT Recover terminates. However, if the output from an index or table space recovery is to be directed to a non-DB2 data set, NGT Recover does not issue any DB2 commands.
Restoring initial status

NGT Recover provides the RDB2STAT utility parameter, which determines whether DB2 objects should be returned to their original status when a recovery job is complete. If you specify RDB2STAT(YES) (or default to the RDB2STAT installation option when it is set to YES), NGT Recover handles the space as follows:

- Starts the space with ACCESS(RW) status if the space was in RW status initially
- Starts the space with ACCESS(RO) if the space was in RO status
- Will not start the space if the space was in STOP, STOPE, LSTOP, or STOPP status
- Starts the space with START ACCESS (FORCE) if the space was in DEFER, LPL status, GRECP status, or internal ERROR RANGE status

You may also use RDB2STAT(RW) and RDB2STAT(RO) on the EXEC statement to request that DB2 objects be restored to a status of RW or RO, regardless of their initial status. For a description of the RDB2STAT parameter, see “RDB2STAT override parameter (rdb2Stat)” on page 317.

If you do not specify RDB2STAT, NGT Recover defaults to the value that is specified for the RDB2STAT installation option.

Serialization and concurrency issues

NGT Recover allows you to run the utility concurrently with other NGT Recover runs, with other utilities, and with applications that access target table spaces.

Concurrency with IBM DB2 utilities

NGT Recover uses a DB2 command interface to obtain the initial DB2 status of each space involved in the recovery and determine whether an IBM DB2 utility is already operating against that space. If the DB2 status of a space indicates that an IBM DB2 utility is operating against the space, NGT Recover terminates. Otherwise, NGT Recover issues DB2 commands to stop all of the spaces to ensure exclusive use of the space and prevent any IBM utility from operating against the space until the recovery is complete. STOP and START commands are issued by partition as appropriate.

For more information, see “Table space and index space status” on page 70.
Concurrency with other BMC utilities for DB2

All BMC utility products use the BMCUTIL table to control the use of utility IDs, which identify executions of BMC utilities. Only one utility with a specific utility ID can run at one time, and NGT Recover terminates when it is started for a space against which another BMC utility is already running.

For more information about this table, see “BMCUTIL table” on page 623.

BMC utility products use the BMCSYNC table to coordinate access to DB2 objects. DB2 objects that participate in a BMC utility job are registered in the BMCSYNC table. When each object is registered, the registering utility assigns a share level to control access to that object from other BMC utilities. For partitioned DB2 spaces, registration is performed at the partition level. For more information about this table, see “BMCSYNC table” on page 614.

The BMCSYNC table allows multiple BMC utilities (or multiple instances of a single utility) to operate concurrently on different partitions of a DB2 space if no nonpartitioning indexes are involved. In addition, some BMC utilities can operate concurrently on the same object or partition. For information about which products can operate concurrently, see Table 4 on page 75.

The "Access level" column in Table 4 on page 75 refers to the value of the SHRLEVEL column in the BMCSYNC table. The level can be one of the following values:

- S indicates shared access. Any other utility that registers with shared access (S) can run against the object.
- X indicates exclusive access. No other utility can run against the object.
- A blank value indicates that no status is requested and any other utility can run against the object.

Table 4: Running BMC products concurrently

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK PLUS</td>
<td>S</td>
<td>None</td>
</tr>
</tbody>
</table>
| NGT Copy                 | S or blank       | - If you specify COPY IMAGECOPY, NGT Copy registers the object with no access status (blank).  
                           |                  | - In all other cases, NGT Copy registers the object with shared access (S).              |
| DASD MANAGER PLUS (BMC STATS) | S                | None                                                                                   |
### Product Access level Additional information

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOADPLUS</td>
<td>X</td>
<td>If you specify PART, LOADPLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions concurrently with this job.</td>
</tr>
</tbody>
</table>
| NGT Recover       | X, S, or blank | - Under the following conditions, NGT Recover registers an object with shared access (S):  
|                   |              |   - If an index is being rebuilt, the table space for that index is registered with shared access if that table space is not also recovered in the same job.  
|                   |              |   - A table space partition is registered with shared access if the keys for that partition are unloaded with a RECOVER UNLOADKEYS operation.  
|                   |              | - If you specify the following commands or options, NGT Recover registers the object with no access status (blank):  
|                   |              |   - The ACCUM command  
|                   |              |   - OUTCOPY ONLY  
|                   |              |   - INDEP OUTSPACE  
|                   |              | - In all other cases, NGT Recover registers the object with exclusive access (X).  |
| RECOVERY MANAGER  | S            | None                                                                                   |
| REORG PLUS        | X            | If you specify PART, REORG PLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions concurrently with this job. |
| UNLOAD PLUS       | S            | None                                                                                   |

### Concurrency with other NGT Recover executions

NGT Recover allows a high degree of concurrency between NGT Recover jobs. You can recover different partitions of the same table space or index space concurrently in different NGT Recover jobs.

However, the following scenarios are not allowed in NGT Recover:

- Submit two NGT Recover jobs concurrently to recover all of the data sets of a space or the same data set of a space
Recover a table space partition and unload keys from the same partition in two
different concurrent jobs

The utility issues a warning if you run more than one job that specifies RECOVER
UNLOADKEYS for the same index and from the same partition or partitions of a
table space.

Concurrency with RECOVER UNLOADKEYS and RECOVER
BUILDINDEX

A special case of concurrency is one or more RECOVER UNLOADKEYS runs
followed by a RECOVER BUILDINDEX run. Because these operations can be run in
separate jobs, updates to the table space must be prevented between the completion
of RECOVER UNLOADKEYS and the beginning of RECOVER BUILDINDEX to
guarantee the integrity of the data.

For this reason, RECOVER UNLOADKEYS leaves the table space in RO (read only)
status. NGT Recover also adds a row to the BMCSYNC table for each partition from
which keys have been unloaded; the utility ID is blank and SHRLEVEL is "S". This
status prevents any other NGT Recover run from recovering the table space before
the row is removed by the RECOVER BUILDINDEX run.

In the same way, the RECOVER UNLOADKEYS run adds a row to the BMCSYNC
table for each nonpartitioned index unloaded; the utility ID is blank and the
BMCSYNC row is shared among all of the RECOVER UNLOADKEYS jobs for the
same index. The RECOVER BUILDINDEX job deletes the BMCSYNC rows with
blank utility IDs that were left by the corresponding RECOVER UNLOADKEYS jobs.
The RECOVER BUILDINDEX job also checks the row for each index first to verify
that all partitions have been unloaded, and issues a warning message if the row is
not found. The job terminates with an error if it finds the row with indications that
not all partitions have been unloaded.

For specific examples of concurrent NGT Recover jobs, see “Using multiple
commands” on page 88.

Recovery using a non-DB2 backup

NGT Recover allows you to recover a space without the use of a DB2 image copy.
You can accomplish this task by restoring the space from a non-DB2 backup and
then by running an NGT Recover job that uses either the LOGAPPLY ONLY or
LOGONLY option. An entire disk pack or a single space can be recovered in this
manner.

When you use the LOGAPPLY ONLY option, you must specify FROMLOGPOINT.
You must select this log point with care to ensure that it represents a point prior to
the non-DB2 backup at which all buffers were externalized. A good rule is to select a
log point that is three checkpoints prior to the time of the backup. A quiesce point prior to the time of the backup is also an acceptable log point. Unpredictable results may occur if the selected value of FROMRBA or FROMLOGPOINT is in a time when the space is in an exceptional state such as error range status or deferred status.

When a value for FROMLOGPOINT is established, the space or disk pack should be restored from the non-DB2 backup. The restoration is followed by the NGT Recover run that uses LOGAPPLY ONLY and the selected value of FROMLOGPOINT.

The LOGONLY option uses the log point that is recorded in the space as the starting log point.

**Recovery using cumulative incremental image copies**

By using NGT Copy, you can merge incremental image copies while retaining the previous incremental copies in SYSIBM.SYSCOPY (BMCXCOPY for incremental index copies) by specifying CUMULATIVE YES KEEP YES. If you specify these options for an incremental copy, the previous copy is registered as ICTYPE=i. This copy is referred to as a cumulative copy. For incremental copies of table spaces, to make this copy available for DB2 RECOVER you must run the NGT Copy RECALL job to set ICTYPE=I.

If a space is recovered to the current point, NGT Recover ignores incremental copies with ICTYPE=i. If a space is recovered to a prior point in time, NGT Recover recognizes any incremental copies with ICTYPE=i that would be useful for the recovery.

For example, if you are recovering to a cumulative incremental copy, NGT Recover treats the cumulative incremental copy as a true copy (ICTYPE=I). If you are recovering to a specific log point and a cumulative incremental copy exists prior to that log point, NGT Recover merges the cumulative incremental copy with the full copy before applying the log. Fallback processing also recognizes cumulative incremental copies and uses those copies as necessary.

For more information about the CUMULATIVE, KEEP, and RECALL options, see the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*.

**Checkpoints for NGT Recover restart**

You can facilitate restarting an NGT Recover job that fails to complete successfully by specifying the checkpoints to be taken in the original run. Processing phases are
not re-executed during the restarted job if checkpoints were taken. The end of a phase is recorded in the BMCSYNC table, and a job can be restarted from the last recorded checkpoint.

Two installation options, CHECKPT and CHECKINT, determine when checkpoints are taken for NGT Recover jobs. In addition, you can override the CHECKINT installation option at run time by using the NGT Recover OPTIONS command with the CHECKINT keyword.

**Checkpoint installation options**

You can specify the CHECKPT installation option to take no checkpoints, or to take checkpoints at the end of each phase.

When you specify checkpoints to be taken, you can prevent taking unnecessary checkpoints by specifying a value for the CHECKINT (checkpoint interval) installation option, which specifies the minimum number of minutes that must elapse before the next checkpoint is taken. To decide on values for CHECKPT and CHECKINT, you must balance the cost of taking checkpoints against the time lost redoing work when an NGT Recover execution fails and must be restarted. For more information about these installation options, see “NGT Recover installation options” on page 573.

**Checkpoint override parameter**

At run time, you can override the value for the CHECKPT installation option by specifying a value for the checkpoint override parameter in your NGT Recover JCL.

Use the following guidelines:

- Specify CHECKPT(NO) (no check points) for short recoveries that you do not mind rerunning when necessary.

- Specify CHECKPT(PHASE) (checkpoints at the end of each phase) for longer running jobs when it becomes costly to rerun the entire job.

As with the CHECKPT installation option, the setting of CHECKINT controls the frequency of the taking of checkpoints when you use the checkpoint override parameter.

For more information about the checkpoint override parameter, see “Building and running NGT Recover jobs” on page 311.
Storage group-defined data sets

For a storage group-defined space, when NGT Recover defines the underlying data set, NGT Recover uses the following algorithm:

1. Obtain volume information for the storage group definition from SYSIBM.SYSVOLUMES in the DB2 system catalog.

2. If the data set is currently cataloged, obtain the volume information from the operating system catalog entry. Arrange to favor those volumes if they are still part of the STOGROUP definition.

3. Determine whether the data set is SMS managed by calling IGWAQSMS (DFP 3.2) or IGWASYS (DFP 3.3 or later).

4. If the data set exists, delete it (unless REDEFINE NO is specified). If it is not on the volume indicated by the operating system catalog entry, delete the data set with the NOSCRATCH option of IDCAMS. If REDEFINE YES NOSCRATCH is specified, no attempt to scratch the data set is made; it is simply uncataloged with the NOSCRATCH option of IDCAMS.

5. Define the data set. The following pseudo code shows which volume is selected for the initial definition of the data set and any subsequent extend requests that require a new volume. If a define or extend request fails, NGT Recover tries the algorithm again. Volumes on which an extend request fails are internally marked as tried.

```plaintext
IF THE STOGROUP DEFINITION INCLUDES SPECIFIC VOLSERS AND ANY VOLSER UNTRIED
      *DEFINE/EXTEND USING THE NEXT UNTRIED VOLSER
ELSE
  IF THE STOGROUP DEFINITION INCLUDES '*' AND THE DATA SET IS SMS MANAGED
     DEFINE/EXTEND USING '*'
  ELSE
     DEFINE/EXTEND USING AN ORIGINAL VOLSER THAT IS UNTRIED
```
Note
Specific volume serial numbers are used in the order in which they are returned by SQL from SYSIBM.SYSVOLUMES. For more information about how to change the volume ordering, please see the discussion of the STOGROUP USEORDER option in “STOGROUP specification description” on page 112.

If a nonpartitioned space is in multiple data sets, unused data sets are removed even with the REDEF option set to NO.

If the extend request requires a new data set or volume, it will be defined even with the REDEF option set to NO.

If an Instant Snapshot copy is used to recover the object, REDEFINE YES has no effect and the data sets are processed as if REDEFINE NO had been specified.

Primary space allocation

If the PRIQTY value is specified, it is used for primary space allocation. If PRIQTY is not specified, the DSNZPARM TSQTY value is used. If TSQTY is set to 0, then 1 cylinder is the default primary space allocation.

Secondary extents

The SECQTY value is used for secondary extents if DSNZPARM MGEXTSZ is set to NO. If MGEXTSZ is set to YES or SECQTY is not specified, NGT Recover uses a sliding-scale calculation for secondary extents, similar to the method DB2 uses.

Note
For more information about the sliding-scale calculation, see the DB2 for z/OS SQL Reference.
NGT Recover syntax

This chapter provides brief descriptions of the purpose and usage of the commands that you can use with NGT Recover.

For information about using multiple NGT Recover commands, see “Using multiple commands” on page 88.

Note
For quick reference, the syntax diagrams are duplicated without any descriptive text in “NGT Recover syntax diagrams” on page 639.

OPTIONS

The OPTIONS command specifies options that are set globally for use in an NGT Recover job. You can have more than one OPTIONS command statement in a job but a particular specification can only be defined one time. You must place the OPTIONS command statement or statements before any other NGT Recover command statement in your job.

For more information, see “OPTIONS command” on page 103.

OUTPUT

The OUTPUT command specifies options that you can code for the dynamic allocation of output data sets. Options are available for disk data sets and for tape data sets.

For more information, see “OUTPUT command” on page 142.
RECOVER TABLESPACE, RECOVER INDEX, RECOVER INDEXSPACE

The RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands specify one or more table spaces or indexes, or table space or index data set, for recovery. You can recover a single table space, index, or data set to a specific image copy, to a specific log point, or to the current state (by using all of the log records that apply).

Multiple table spaces, indexes, or data sets can be recovered to a specific log point or to the current state. You can have as many RECOVER TABLESPACE, RECOVER INDEX, or RECOVER INDEXSPACE command statements in a job step as necessary. However, unlike the IBM DB2 RECOVER utility, NGT Recover examines all of the command statements in the job step before processing starts so that all log processing can be combined and activities can be optimally scheduled.

**Note**

If you request simulation mode with the SIMULATE YES option, which requires a valid BMC Recovery Management for DB2 solution password, you must use SIMRCVR TABLESPACE, SIMRCVR INDEX, and SIMRCVR INDEXSPACE instead of RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE.

For more information, see “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE option descriptions” on page 167.

**Note**

This book assumes that RECOVER INDEX uses copies, logs, or both. However, NGT Recover can interpret the RECOVER INDEX syntax as REBUILD INDEX if the option INDEXLOG is set to NO (the default). INDEXLOG AUTO could have the same effect in some circumstances. INDEXLOG is set by using the INDEXLOG installation option or the INDEXLOG option on the OPTIONS command (see “INDEXLOG=NO” on page 584 and “INDEXLOG” on page 124).

REBUILD INDEX

The REBUILD INDEX command specifies that one or more indexes on tables in a single table space are to be rebuilt from the data in the table space. If the index is a partitioned index on a partitioned table space and you do not specify a partition, NGT Recover rebuilds all partitions of a partitioned index. You can specify a single partition of the index.
You can include multiple REBUILD INDEX command statements in a single NGT Recover execution, along with RECOVER TABLESPACE and RECOVER UNLOADKEYS command statements. All command statements are examined and processed as efficiently as possible.

For more information, see “REBUILD INDEX” on page 215.

**Note**

If you request simulation mode with the SIMULATE YES option, which requires a valid BMC Recovery Management for DB2 solution password, you must use SIMRBLD INDEX rather than REBUILD INDEX.

---

## RECOVER UNLOADKEYS

The RECOVER UNLOADKEYS and RECOVER BUILINDEX commands work together but in separate executions. RECOVER UNLOADKEYS extracts from the table space the index keys that RECOVER BUILINDEX uses to build nonpartitioned indexes on a partitioned table space.

RECOVER UNLOADKEYS specifies nonpartitioned indexes on a partitioned table space and specifies one partition of the table space from which the keys for those indexes are extracted. The keys are written to data sets which are specified in a RECOVER BUILINDEX command statement in a subsequent job. The keys within these data sets are sorted during the RECOVER UNLOADKEYS processing or the RECOVER BUILINDEX.

To extract keys from multiple partitions, you use multiple RECOVER UNLOADKEYS command statements. You can include these with RECOVER TABLESPACE and REBUILD INDEX command statements.

For more information, see “RECOVER UNLOADKEYS” on page 237.

**Note**

If you request simulation mode with the SIMULATE YES option, which requires a valid Recovery Management for DB2 solution password, you must use SIMRCVR UNLOADKEYS rather than RECOVER UNLOADKEYS.

---

## RECOVER BUILINDEX

The RECOVER BUILINDEX command specifies the rebuilding of one or more nonpartitioned indexes on a partitioned table space by using data sets that are
created in earlier RECOVER UNLOADKEYS runs. You can include only one RECOVER BUILDINDEX command statement in the NGT Recover execution; no other NGT Recover command statements are allowed.

For more information, see “RECOVER BUILDINDEX” on page 246.

**Note**

If you request simulation mode with the SIMULATE YES option, which requires a valid BMC Recovery Management for DB2 solution password, you must use SIMRCVR BUILDINDEX rather than RECOVER BUILDINDEX.

---

**LOGSCAN**

The LOGSCAN command provides information about the number and size of log records required for recovery. You can use this information to size the log sort. NGT Recover prints a report describing the log records for each object that is named in the LOGSCAN.

For more information, see “LOGSCAN” on page 253.

---

**MIGRATE**

The MIGRATE command allows you to move data from one or more table spaces to another. The table spaces can be on the same subsystem, or they can be on different subsystems, as long as the two subsystems share DASD. Image copies, and optionally DB2 log, from the source system are used to create the target image.

For more information, see “MIGRATE” on page 270.

---

**IMPORT**

The IMPORT command allows you to move data, by use of image copies, from one or more table spaces to another.

For more information, see “IMPORT” on page 290.
Adding comments

You can code comments in NGT Recover command statements by bracketing the comment with /* at the beginning and */ at the end. For example:

```plaintext
//SYSIN DD*
/*Recover production table space*/
RECOVER TABLESPACE PRODDB.PRODTS
TORBA LASTQUIESCE /*prior to last batch update*/
```

**Note**
When coding /*, avoid column 1 because this string could terminate prematurely your SYSIN.

You can also code comments by preceding information with a double hyphen (--). A comment that is started with the hyphens runs to the end of the line. You can place the double hyphen in column 1 through column 70. Do not break the double hyphen across a line. An example follows:

```plaintext
//SYSIN DD*
-- Recover production table space
RECOVER TABLESPACE PRODDB.PRODTS
TORBA LASTQUIESCE --prior to last batch update
```

All characters inside comments, both those specified with /* and those specified with a double hyphen, are ignored.

Use of long names

NGT Recover supports long names up to 128 bytes in length for the following identifiers:

- Table names
- Index names
- Creator names
- STOGROUP names

The only long names that you can use in the SYSIN file are creator names and index names. NGT Recover reads the long names from SYSIN, parses them, saves them in control blocks, and displays long names in output messages. Output messages can contain long creator, index, and table names, which may cause messages to require multiple lines.

In SYSIN, long names must be in columns 1 to 72. Columns 73 to 80 are ignored. You can split long names in SYSIN across lines. If a name is split across lines, the name must continue to column 72 with no embedded spaces, and the remainder of the name must start in column 1 on the next line.
Support for Unicode names

NGT Recover provides support for Unicode names as follows:

- Unicode is not supported in the SYSIN file. However, NGT Recover can process spaces that contain tables or indexes with Unicode names. (Spaces cannot contain Unicode names in DB2.) NGT Recover processes spaces with STOGROUP names, index names, and index creator names in Unicode.

NGT Recover commands with wild cards do not include objects that match the pattern but contain Unicode characters that are not translatable to EBCDIC in the wild card position. This result is because SYSIN is EBCDIC and wild card processing is done in EBCDIC.

- In output files, NGT Recover displays Unicode names for index names (not index space names) or table names that do not translate to EBCDIC as a UTF-8 (Unicode Transformation Format, 8-bit encoding form) representation in hexadecimal delimited by angle brackets (< >).

Figure 2 on page 88 shows an example of how NGT Recover represents Unicode in SYSPRINT.

Figure 2: Unicode representation in the NGT Recover output file

Using multiple commands

Multiple RECOVER TABLESPACE, RECOVER INDEX, RECOVER INDEXSPACE, REBUILD INDEX, and RECOVER UNLOADKEYS commands are allowed in a single job. However, because NGT Recover attempts to coordinate all requested work, certain restrictions apply. Only a single RECOVER BUILDINDEX command is allowed and, when present, must be the only command for the job step. The RECOVER BUILDINDEX command supports only one table space but does support multiple indexes on that table space.

Note

NGT Recover does not support commands in the input data set that execute other utilities.
General restrictions for multiple commands

The following general restrictions apply when you use multiple command statements:

- No space may be named or implied for recovery in an NGT Recover job more than once. For the purposes of this restriction, the data sets of a table space are considered separate objects. For example, the following command statements would be incorrect if AFRIND1 was an index on a table in table space AFRDB.AFRTS:

```
REBUILD INDEX (AFRIND1)
REBUILD INDEX(ALL) TABLESPACE AFRDB.AFRTS
```

However, the following command statements are correct:

```
REBUILD INDEX (AFRIND1) PART 1
REBUILD INDEX (AFRIND1) PART 2
REBUILD INDEX (AFRIND2)
```

If RECOVER UNLOADKEYS command statements are present for any partition of a table space, REBUILD INDEX command statements on nonpartitioned indexes for the same table space are not valid. However, REBUILD INDEX command statements for the partitioned index on the space are valid.

For example, the following command statements are incorrect if AFRIND2 is a nonpartitioned index:

```
REBUILD INDEX (AFRIND2) TABLESPACE AFRDB.AFRTS
RECOVER UNLOADKEYS (AFRIND3) PART 1 TABLESPACE AFRDB.AFRTS
```

Because the first command statement requires that all partitions of the table space be read to extract keys for the nonpartitioned index, you should perform one of the following actions:

- Unload keys for both indexes in one or more jobs and follow these jobs with a RECOVER BUILDINDEX job
- Recover both indexes in one job by using REBUILD INDEX

The following command statements are correct:

```
REBUILD INDEX (AFRIND1) PART 1
RECOVER UNLOADKEYS (AFRIND2, AFRIND3) PART 1
```

This execution rebuilds the partitioned index while extracting the keys for nonpartitioned indexes from partition 1 of the space. NGT Recover processes these two command statements concurrently.

- RECOVER BUILDINDEX, when present, must be the only command in the job step.
- The ANALYZE option may be on any command statement and overrides the default (ANALYZE YES). If you specify the option on more than one NGT Recover command statement, the value (YES, NO, or ONLY) must be the same in all cases.
When you are recovering a cloned object, all related objects named in the job must also be clones. For example, the following command statements are incorrect:

```
RECOVER TABLESPACE AFRDB.AFRTS CLONE
RECOVER INDEX(ALL) TABLESPACE AFRDB.AFRTS
```

### Restrictions for STOGROUP-defined objects

When you want to recover multiple STOGROUP-defined objects with some objects reallocated and others not reallocated (by using the REDEFINE option), you must use one command statement to reallocate objects and a second command statement for those objects that are not reallocated, as follows:

- For table space recovery, use one `RECOVER TABLESPACE` command statement for table spaces that are reallocated and a second `RECOVER TABLESPACE` command statement for table spaces that are not reallocated. For more information, see “REDEFINE” on page 212.

- To rebuild an index space by using `REBUILD INDEX`, use one command statement for index spaces that are reallocated and a second command statement for index spaces that are not reallocated. For more information, see “REDEFINE” on page 233.

- For index space recovery by using `RECOVER BUILDINDEX`, use one command statement for index spaces that are reallocated and a second command statement (in a separate execution) for index spaces that are not reallocated. For more information, see “REDEFINE” on page 252.

### General information about SORTDEV and SORTNUM

You can specify `SORTDEV` and `SORTNUM` in the installation options, or with the following NGT Recover commands:

- `OPTIONS`
- `RECOVER TABLESPACE`, `RECOVER INDEXSPACE`, or `RECOVER INDEX`
- `RECOVER UNLOADKEYS`
- `REBUILD INDEX`

`SORTDEV` and `SORTNUM` values override other values as follows:

- Values on the preceding `RECOVER` commands or the `REBUILD INDEX` command override the values on the `OPTIONS` command
- Values on the `OPTIONS` command override the values in the installation options module
If you specify SORTNUM with the OPTIONS, RECOVER, or REBUILD commands, and you do not specify SORTDEVT with any of the commands or you define SORTDEVT in the installation options module, NGT Recover ignores SORTNUM.

**Restrictions for key work data sets, SKEYDDN, SORTNUM, and SORTDEVT**

You may have multiple REBUILD INDEX command statements, RECOVER UNLOADKEYS command statements, or both. Restrictions on the key work data sets and the SORTNUM and SORTDEVT options of these statements are designed to allow the most efficient processing of all command statements. The restrictions are determined by whether the table space for the indexes is partitioned or nonpartitioned and, if partitioned, whether nonpartitioned indexes are being rebuilt or whether RECOVER UNLOADKEYS is specified.

**Restriction for nonpartitioned table space recoveries**

You do not need to specify multiple REBUILD INDEX command statements for one nonpartitioned table space, but with NGT Recover you can specify multiple statements. However, because NGT Recover unloads the keys concurrently for all of the indexes, the key work data sets and the SORTDEVT and SORTNUM options must be the same on all statements for one table space.

**Restrictions for partitioned table spaces**

Restrictions for partitioned table spaces vary according to whether you are rebuilding a partitioned or nonpartitioned index and whether you specify RECOVER UNLOADKEYS.

**Partitioned index rebuild**

If you specify only partitioned index recovery, no restrictions exist on the key work data sets or the SORTNUM and SORTDEVT options. The following examples are allowed:

```
REBUILD INDEX (AFRIND1) PART 1 WORKDDN KEYS
REBUILD INDEX (AFRIND1) PART 2 NOWORKDDN
REBUILD INDEX (AFRIND1) PART 1 WORKDDN KEYS
REBUILD INDEX (AFRIND1) PART 2 WORKDDN KEYS
```

**Note**

You can use WORKDDN only with MAXKSORT 1.
Nonpartitioned index rebuild

When you specify REBUILD INDEX for a nonpartitioned index, the key work data sets and the SORTDEVT and SORTNUM options must agree on all REBUILD INDEX command statements for the table space, including those for all of a partitioned index or parts of a partitioned index. This agreement allows NGT Recover to extract keys for all indexes from each partition during one pass of the table space.

RECOVER UNLOADKEYS specified

When you specify RECOVER UNLOADKEYS, a data set is created to hold the keys for a subsequent job that builds the index. The keys for all partitions of a table space within the step are required to be in one data set that is designated by the SKEYDDN ddname. Generally, if you run RECOVER UNLOADKEYS jobs for every table space partition, you can code one of the following command statements (in each job):

\[
\text{RECOVER UNLOADKEYS(ALL) TABLESPACE...PART \( nn \)}
\]

\[
\text{RECOVER UNLOADKEYS (indexName1,...,indexNamen)PART \( nn \)}
\]

If you specify multiple RECOVER UNLOADKEYS command statements for the table space, you must use the same SKEYDDN option on these command statements. If other REBUILD INDEX or RECOVER UNLOADKEYS command statements reference other table spaces or other partitions, the keys for different table spaces must be unloaded to different data sets. Name these data sets with the SKEYDDN option.

For example, you could code the following command statements:

\[
\text{RECOVER TABLESPACE AFRDB.AFRTS1 DSNUM 1}
\]

\[
\text{RECOVER TABLESPACE AFRDB.AFRTS2 DSNUM 1}
\]

\[
\text{RECOVER UNLOADKEYS (ALL) TABLESPACE AFRDB.AFRTS1 PART 1}
\]

\[
\text{REBUILD INDEX (AFRIND1) TABLESPACE AFRDB.AFRTS1 PART 1}
\]

\[
\text{RECOVER UNLOADKEYS (ALL) TABLESPACE AFRDB.AFRTS2 PART 1}
\]

\[
\text{SKEYDDN TS2K}
\]

\[
\text{REBUILD INDEX (AFRCLUS) TABLESPACE AFRDB.AFRTS2 PART 1}
\]

The keys for the nonpartitioned indexes of AFRDB.AFRTS1 are directed to the default DDNAME SKEY. The keys for the nonpartitioned indexes of AFRDB.AFRTS2 are directed to the DDNAME TS2K. SKEY and TS2K are ddnames, not prefixes. The keys for the partitioned indexes AFRIND1 and AFRCLUS are directed to NOWORKDDN by default.

Alphabetical listing of NGT Recover options

NGT Recover options are listed in the following table, alphabetized by NGT Recover command, and within the command by option name.
IMPORT command options

- ANALYZE
- AUX
- CHANGE ... FROM...TO
- CLONE
- DEFINE
- DSNUM
- EXCLUDE
  — For OBJECTSET specification
  — For USING specification
- INDEP OUTSPACE
- INDEXES
- MODEL
- NOCOPYPEND
- NOSCRATCH
- OBJECTSET
- OUTCOPY
- OUTCOPYDDN
- REBUILD
- RECOVERYDDN
- REDEFINE
- RESET
- REUSE
- RICHK
- SYNC
- TABLESPACE
- UPDATE VERSIONS
- USING

For information about these options, see “IMPORT” on page 290.

LOGSCAN command options

- AUX
- BACKOUT
- CLONE
- DSNUM
- FROMLOGPOINT
  — LASTARCHQ
  — LASTCOPY
LASTQUIESCE
LASTSHUTDOWN
X'logPoint'

FROMRBA
— LASTARCHQ
— LASTCOPY
— LASTQUIESCE
— LASTSHUTDOWN
— X'logPoint'

INDEX
— For single index specification
— For multiple index specification

INDEXES (or INDEX)

INDEXSPACE
— For single index specification
— For multiple index specification

LOGAPPLY ONLY
LOGONLY
OBJECTSET
TABLESPACE

TOLOGPOINT
— LASTARCHQ
— LASTCOMMONQ
— LASTQUIESCE (relativeGenerationNumber)
— LASTSHUTDOWN
— X'logPoint'

TORBA
— LASTARCHQ
— LASTCOMMONQ
— LASTQUIESCE (relativeGenerationNumber)
— LASTSHUTDOWN
— X'logPoint'

For information about these options, see “LOGSCAN” on page 253.

MIGRATE command options

— ANALYZE
— CHANGE ... FROM...TO
— CLONE
- DSNUM
- EXCLUDE
  - For OBJECTSET specification
  - For USING specification
- FROMLOGPOINT
  - LASTARCHQ
  - LASTCOPY
  - LASTQUIESCE
  - LASTSHUTDOWN
  - X'logPoint'
- FROMRBA
  - LASTARCHQ
  - LASTCOPY
  - LASTQUIESCE
  - LASTSHUTDOWN
  - X'logPoint'
- INDEP OUTSPACE
- LOGAPPLY ONLY
- LOGONLY
- LOGSORT
- MODEL
- NUMREC
  - ABS
  - AVGRECSZ
  - CALC
  - EST
  - NOEST
- OBJECTSET
- RESET
- SORTDEVT
- SORTNUM
- TABLESPACE
- TOCOPY LASTCOPY
- TOLOCATION
- TOLOGPOINT
  - LASTARCHQ
  - LASTCOMMONQ
  - LASTQUIESCE (relativeGenerationNumber)
  - LASTSHUTDOWN
  - LOGMARK logMarkName (logMarkGeneration)
— X'logPoint'

■ TORBA
  — LASTARCHQ
  — LASTCOMMONQ
  — LASTQUIESCE (relativeGenerationNumber)
  — LASTSHUTDOWN
  — LOGMARK logMarkName (logMarkGeneration)
  — X'logPoint'
■ USING

For information about these options, see “MIGRATE” on page 270.

OPTIONS command options

■ ANALYZE
■ AUTOSIZE
■ AUX
■ BACKOUT
■ CHECKINT
■ DATAMVR
■ DEFINE
■ EARLYCAT
■ EARLYRECALL
■ INDEXLOG
■ IXRECP
■ KEYSORT
■ KSORTSHARE
■ LOGPOINT X'logPoint' or LOGPOINT LOGMARK logMarkName (logMarkGeneration)
■ LOGSORT
■ MAXDRIVES
■ MAXKSORT
■ MAXLOGS
■ MAXLSORT
■ NOEARLYCAT
■ NOEARLYRECALL
■ NOSYSLGRNG
■ NUMREC
  — ABS
  — AVGRECSZ
  — CALC
— EST
— NOEST
■ ON ERROR ANY CONTINUE
■ OUTCOPY
— ASCODED
— BYPART
■ REBUILDSCOPE
■ RECOVERSCOPE
■ RECOVERYPONT
■ RESINV
■ RESOURCE SELECTION
— ACCUMS
— COPIES
— LOGS
■ RICHK
■ SIMULATE YES
■ SMCORE
■ SNAP
■ SORTDEVT
— For KEYSORT
— For LOGSORT
■ SORTDIAG
■ SORTDYN
■ SORTNUM
— For KEYSORT
— For LOGSORT
■ STOGROUP... USEORDER
■ TIMESTAMP
■ TRTCH
■ URIDDDN
■ USEACCUM
■ USEHDROBIDS
■ WTOR
■ XBMID
■ ZIIP

For information about these options, see “OPTIONS command” on page 103.
**OUTPUT command options**

- CATLG
- DATACLAS
- DSNAME
- EATTR
- EXPDT
- MAXPRIM
- MGMTCLAS
- MODELDCB
- RETPD
- SPACE
- STACK
- STORCLAS
- TRTCH
- UNIT
- UNITCNT
- VOLCNT
- VOLUMES

For information about these options, see “OUTPUT command” on page 142.

**REBUILD INDEX and SIMRBLD INDEX command options**

- ANALYZE
- AUX
- CLONE
- DEFINE
- INDEP INTABLESPACE
- INDEP OUTSPACE
- MODEL
  - For INDEP INTABLESPACE
  - For INDEP OUTSPACE
- NOSCRATCH
- NUMREC
  - ABS
  - CALC
  - EST
  - NOEST
- NOWORKDDN
- OBJECTSET
PART
REDEFINE
REUSE
SCOPE
— ALL
— PENDING
— STATUS(status1, status2, …)
SORTDEVT
SORTNUM
TABLESPACE
WORKDDN

For information about these options, see “REBUILD INDEX” on page 215.

RECOVER BUILDINDEX and SIMRCVR BUILDINDEX command options

ANALYZE
CLONE
INDEP OUTSPACE
MODEL
NOSCRATCH
REDEFINE
REUSE
SKEY
SKEYDDN
TABLESPACE

For information about these options, see “RECOVER BUILDINDEX” on page 246.

RECOVER INDEX, RECOVER INDEXSPACE, RECOVER TABLESPACE, SIMRCVR INDEX, SIMRCVR INDEXSPACE, and SIMRCVR TABLESPACE command options

ANALYZE
AUX
BACKOUT
CLONE
DBID
DEFINE
DROPRECOVERY
Alphabetical listing of NGT Recover options

- DSNAME
- DSNUM
  - For INDEX
  - For INDEXSPACE
  - For TABLESPACE
- DSSIZE
- ENCRYPTED
- FROMLOGPOINT
  - LASTARCHQ
  - LASTCOPY
  - LASTQUIESCE
  - LASTSHUTDOWN
  - X'logPoint'
- FROMRBA
  - LASTARCHQ
  - LASTCOPY
  - LASTQUIESCE
  - LASTSHUTDOWN
  - X'logPoint'
- INCOPY FULL
- INCR
- INDEP OUTSPACE
- INDEVT
- INDEX
  - For single index specification
  - For multiple index specification
- INDEXES (or INDEX)
- INDEXSPACE
  - For single index specification
  - For multiple index specification
- INLINE
- INLOG LOGPOINT
- INLOG RBA
- INSEQNO
- INVOLUME
- LASTCOPY (relativeGenerationNumber)
- LOCALSITE
- LOGAPPLY ONLY
- LOGONLY
- LOGPOINT
- LOGSORT
- MODEL
- NOCOPYPEND
- NOSCRATCH
- NOWORKDDN
- NUMREC
  - ABS
  - AVGRECSZ
  - CALC
  - EST
  - NOEST
- OBID
- OBIDXLAT
- OBJECTSET
- OUTCOPY
- OUTCOPYDDN
- OUTCOPYDSN
- PIECESIZE
- PSID
- RBA
- RECOVERYDDN
- RECOVERYDSN
- RECOVERYSITE
- REDEFINE
- REGISTER
- RESET
- RESETRTS
- REUSE
- RICHK
- SCOPE
  - ALL
  - PENDING
  - STATUS(status1, status2, …)
  - UPDATED
- SHRLEVEL
- SNAPSHOT
- SORTDEVT
- SORTNUM
- SPACE
- TABLESPACE
— For table space specification
— For multiple index specification

■ TOCOPY
■ TOLOGPOINT
— LASTARCHQ
— LASTCOMMONQ
— LASTQUIESCE (relativeGenerationNumber)
— LASTSHUTDOWN
— LOGMARK logMarkName (logMarkGeneration)
— X’logPoint

■ TORBA
— LASTARCHQ
— LASTCOMMONQ
— LASTQUIESCE (relativeGenerationNumber)
— LASTSHUTDOWN
— LOGMARK logMarkName (logMarkGeneration)
— X’logPoint

■ TOSEQNO
■ TOVOLUME
■ TRANSFORM
■ UPDATE VERSIONS

For information about these options, see “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands” on page 151.

RECOVER UNLOADKEYS and SIMRCVR UNLOADKEYS command options

■ ANALYZE
■ CLONE
■ INDEP INTABLESPACE
■ MODEL
■ NUMREC
— ABS
— CALC
— EST
— NOEST
■ PART
■ SKEY
■ SKEDDN
OPTIONS command

This section describes the syntax of the OPTIONS command and its options.

Note

OPTION is also accepted in place of OPTIONS in NGT Recover syntax.

- SORTDEVT
- SORTNUM
- TABLESPACE

For information about these options, see “RECOVER UNLOADKEYS” on page 237.
OPTIONS syntax

The following figures show the syntax of the OPTIONS command.

Figure 3: OPTIONS syntax diagram

1 The default ANALYZE ONLY runs
OPTIONS command

- DEFINE
  - NO
  - YES
- RESOURCE SELECTION
- LOGS
  - ACT1
  - ACT2
  - ARC1
  - ARC2
- ACCUMS
  - ACT1
  - ACT2
  - ARC1
  - ARC2
- COPIES
  - LP
  - LB
  - RP
  - RB
  - SB
  - FC

- BACKOUT
  - AUTO
  - NO
  - YES
- INDEXLOG
  - YES
  - IXRECP
  - NO
  - TRTCH
  - NONE
    - COMP
    - NOCOMP
- XBMID ssid or sbmGroup
- ZIIP
  - ENABLED
  - DISABLED
- DATAMVR programName

1 Valid with R+/CHANGE ACCUM only
2 Requires a Recovery Management for DB2 solution password
Figure 4: OPTIONS syntax—LOGSORT specification
NGT Recover provides options for use with OPTIONS command statements. The options are described in the order in which they are shown in the OPTIONS syntax diagram.

**EARLYRECALL / NOEARLYRECALL**

The EARLYRECALL and NOEARLYRECALL options determine whether NGT Recover starts early retrieval of archived image copies, log data sets, and change accumulation data sets.

EARLYRECALL causes data sets to be scheduled for recall before other activities begin. A maximum number of tasks is specified during installation by using the RCLTSK option. The data sets are recalled in the order in which NGT Recover accesses them. If RCLTSK is set to zero, the EARLYRECALL option is not allowed. If ANALYZE YES or ANALYZE NO is specified, EARLYRECALL is the default.

NOEARLYRECALL prevents NGT Recover from recalling the data sets until they are accessed or allocated. If ANALYZE ONLY is specified, NOEARLYRECALL is the default. Including EARLYRECALL in an ANALYZE ONLY job causes the data sets that are needed for recovery to be recalled.
**EARLYRECALL/NOEARLYRECALL**

EARLYRECALL/NOEARLYRECALL does not apply to table space or index space data sets. If such data sets are migrated, they are recalled when allocated.

**EARLYCAT / NOEARLYCAT**

The EARLYCAT and NOEARLYCAT options determine when NGT Recover verifies that cataloged data sets are listed in the operating system catalog. The default value is EARLYCAT.

EARLYCAT enables NGT Recover to verify during the ANALYZE phase that cataloged data sets are listed in the operating system catalog. If the data sets do not exist, NGT Recover determines not to use the data sets during the ANALYZE phase. If other resources are available that allow the recovery to complete normally, they are used.

NOEARLYCAT causes NGT Recover to allow dynamic allocation to determine whether data sets actually exist in the operating system catalog. If a data set does not exist, an error occurs and normal error recovery takes over at that point.

**LOGSORT work data set specification description**

NGT Recover sorts the log and merges the sorted records with copies and change accumulation files. Sorting the log records allows efficient merging of the copies and log records and, if you are rebuilding indexes, permits index keys to be extracted at the same time. By using the LOGSORT specification, you can code values for the SORTDEVT, SORTNUM, and NUMREC options for the sort. You do not need to have log records to use these options (for example, if no updates to the tables and indexes occurred after the most recent copy was made).

If you use the LOGSORT specification values in the OPTIONS command statement, you can also use them in the RECOVER TABLESPACE, RECOVER INDEX, or RECOVER INDEXSPACE command statement; however, this is not necessary.

If you do not use the LOGSORT specification in the OPTIONS command statement and you code multiple RECOVER command statements with the LOGSORT specification on one RECOVER command statement while taking the default on the other RECOVER command statements, the LOGSORT specification values becomes the default for the other command statements. You can code the LOGSORT
specification values on more than one of the RECOVER command statements, but it is not necessary.

**Note**
All LOGSORT parameters must match if they are used on multiple statements in the same job.

LOGSORT specification values are ignored if the LOGSCAN command is present because LOGSCAN does not invoke a sort.

For more about the LOGSORT option, see “Overview of the LOGSORT strategy” on page 458.

NGT Recover invokes BMCSORT, which allocates the required temporary sort work data sets. You can perform one of the following actions:

- Specify SORTNUM and SORTDEVT to direct the allocation
- Specify neither SORTNUM nor SORTDEVT, and let BMCSORT allocate the work data sets according to sort rules
- Specify neither SORTNUM nor SORTDEVT, and provide L001WKnn DD statements in the JCL
- Specify only SORTDEVT, and let BMCSORT determine the number of data sets

If you use L001WKnn DD statements in the JCL, any SORTDEV and SORTNUM specifications that are present are ignored. If you do not use L001WKnn DD statements and do not specify SORTDEVT or SORTNUM, the values that are supplied in the NGT Recover installation options are used.

**SORTDEVT deviceType**

SORTDEVT *deviceType* specifies the device type for the temporary sort work data sets that BMCSORT uses.

If you specify SORTDYN NO (“SORTDYN” on page 114), BMCSORT defaults are used even if you specify a value for SORTDEVT. Otherwise, NGT Recover uses its internal default value, SYSDA, if you do not specify SORTDEVT.

**SORTNUM integer**

The SORTNUM option affects the allocation of sort work files when BMCSORT is allocating your sort work files dynamically. You can specify an integer value of 1 through 255.
When you specify this option, BMCSORT dynamically allocates the number of sort work files that it needs for each sort task up to the maximum that is illustrated in the following formula:

\[
\text{maximum dynamically allocated sort work files} = n - \text{preallocated sort work files}
\]

If you specify \textit{integer} from 1 through 32, \(n\) equals 32. If you specify \textit{integer} greater than 32, \(n\) equals integer.

\[\text{Note}\]
Preallocated sort work files include sort work files that are allocated in your JCL.

If you do not specify a value for \texttt{SORTNUM}, NGT Recover uses the installation option default.

\textbf{NUMREC}

\texttt{NUMREC} specifies the size of the sort file for the log sort routine. The size is defined as an estimate or the exact number of log records to process and the estimated average length of each log record.

For information about performance implications, see “NGT Recover concepts” on page 451.

\textbf{NUMREC CALC}

\texttt{NUMREC CALC} is the default value. Specifying this option allows NGT Recover to estimate the number of log records to be processed by the log sort routine. It estimates that the sort receives one log record per control interval (CI) processed.

\textbf{NUMREC NOEST}

Specifying \texttt{NUMREC NOEST} prevents NGT Recover from passing a file size to the log sort routine.

\textbf{NUMREC EST \textit{integer}}

Use \texttt{NUMREC EST \textit{integer}} to specify an estimated number of log records for the log sort routine. \textit{integer} must be a positive integer.

\textbf{AVGRECSZ \textit{integer}}

Use \texttt{AVGRECSZ \textit{integer}} to specify the average length of the log records to sort. If you do not specify \texttt{AVGRECSZ}, NGT Recover determines its value.
**NUMREC ABS integer**

Use NUMREC ABS integer to specify the exact number of log records for the log sort routine. integer must be a positive integer.

---

**WARNING**

If the integer parameter is a value that does not represent the true number of log records that are passed to the log sort routine, the log sort routine abends. Use extreme caution when you specify NUMREC ABS.

**AVGRECSZ integer**

Use AVGRECSZ integer to specify the average length of the log records to sort. If you do not specify AVGRECSZ, NGT Recover determines its value.

**NOSYSLGRNG**

NOSYSLGRNG causes the SYSIBM.SYSLGRNX table to be bypassed when determining what logs are used during recovery.

NOSYSLGRNG is used automatically when the object to be recovered has been dropped and is being recovered with the DROPRECOVERY option (see “Table space specification” on page 168). If you have a damaged SYSIBM.SYSLGRNX table, you can use the NOSYSLGRNG option to avoid accessing those tables. This option causes all log records to be scanned without regard to log ranges, using SYSCOPY entries and end points to determine what log is needed.

Indexes that are defined as COPY NO do not register SYSIBM.SYSLGRNX entries. Recoveries of COPY NO indexes are effectively NOSYSLGRNG recoveries.

**USEACCUM**

The USEACCUM option indicates whether to access change accumulation files. How USEACCUM works depends on the availability of a valid password for the R+/CHANGE ACCUM for DB2 product or BMC BMC Recovery Management for DB2 solution as shown in the following table:

<table>
<thead>
<tr>
<th>USEACCUM value</th>
<th>With a valid R+/CHANGE ACCUM or Recovery Management password</th>
<th>Without a valid password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not specified</td>
<td>NGT Recover looks for and uses available change accumulation files.</td>
<td>NGT Recover does not look for or use change accumulation files.</td>
</tr>
<tr>
<td>USEACCUM YES</td>
<td>NGT Recover looks for and uses available change accumulation files.</td>
<td>NGT Recover looks for and uses available change accumulation files.</td>
</tr>
<tr>
<td>USEACCUM value</td>
<td>With a valid R+/CHANGE ACCUM or Recovery Management password</td>
<td>Without a valid password</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>USE ACCUM NO</td>
<td>NGT Recover does not look for or use change accumulation files.</td>
<td>NGT Recover does not look for or use change accumulation files.</td>
</tr>
</tbody>
</table>

**USEACCUM YES**

If R+/CHANGE ACCUM is installed, USEACCUM YES is the default.

**USEACCUM NO**

If R+/CHANGE ACCUM is not installed, USEACCUM NO is the default.

**CHECKINT (integer)**

This option specifies the time in minutes between checkpoints.

Use this option to balance frequency of restart points with the overhead incurred when taking restart points. When you specify CHECKINT on the OPTIONS command statement, the value specified overrides the value of the CHECKINT installation option (which defaults to 0).

If the installation option CHECKPT is set to CHECKPT=PHASE, a restart point is taken only if more minutes than those specified by CHECKINT have elapsed since the last restart point was taken. Use CHECKINT 0 (the default value) on the OPTIONS command statement and the CHECKPT installation option set to CHECKPT=PHASE to force a restart checkpoint after every phase.

**STOGROUP specification description**

Use the STOGROUP specification to control the order in which volumes in a STOGROUP are allocated during the execution of NGT Recover. Because a STOGROUP definition in the DB2 catalog has no inherent ordering of the volumes, a STOGROUP specification with the OPTION command statement becomes necessary when you need a specific order.

Allocations for table spaces and index spaces are attempted on volumes in the order specified in the USEORDER clause. Volumes in the DB2 storage group that are not included in the USEORDER clause are used after those explicitly requested and are used in the order that they are retrieved from SYSIBM.SYSVOLUMES.

You can include more than one STOGROUP specification in one OPTION command statement.
If you do not use the STOGROUP specification, the volumes are allocated in the order that they are retrieved from SYSIBM.SYSVOLUMES. For more information, see “Storage group-defined data sets” on page 80.

**STOGROUP storageGroupName**

STOGROUP `storageGroupName` specifies the name of a DB2 storage group. `storageGroupName` must be the name of a storage group that is found in SYSIBM.SYSSTOGROUP. If the storage group is not found in SYSIBM.SYSSTOGROUP, a warning is issued during analysis. If a valid storage group is included in the STOGROUP specification but is not referenced by any of the spaces being recovered, a warning is issued.

**USEORDER volumeSerialNumber**

Left and right parentheses are required around the list. Each `volumeSerialNumber` in the list must be found in the SYSIBM.SYSVOLUMES associated with `storageGroupName`. If a volume serial number that is specified is not found in SYSIBM.SYSVOLUMES, a fatal error occurs during analysis.

In the following OPTION command statement, all table spaces implicated in the recovery that use STOGROUP PRODSG1 will try to find space on the volumes in the order PROD01, PROD02, PROD03, and PROD04. Table spaces that use STOGROUP PRODSG2 will use volumes PROD02, PROD03, PROD04, and PROD01 (in that order).

```
OPTION STOGROUP PRODSG1 USEORDER (PROD01,PROD02,PROD03,PROD04)
STOGROUP PRODSG2 USEORDER (PROD02,PROD03,PROD04,PROD01)
STOGROUP PRODSG3 USEORDER (PROD03,PROD04,PROD01,PROD02)
STOGROUP PRODSG4 USEORDER (PROD04,PROD01,PROD02,PROD03)
```

**SORT specification description**

Use the SORT specification to have greater control over the sort routine.

You can specify values for the parameters that are associated with the SORT specification only on the OPTION command statement in NGT Recover. Your specifications affect all sorts invoked by the recovery. (For more information, see “Managing sort performance” on page 474.)

**Note**

NGT Recover uses BMCSORT as the sort engine. *BMC recommends that you not change any SORT specification option defaults unless necessary.*

**SORTDIAG**

Use the SORTDIAG option to provide sort diagnostic messages.
SORTDIAG NO

SORTDIAG NO is the default and turns off any provision for diagnostic messages.

SORTDIAG YES

SORTDIAG YES turns on diagnostic messages. The messages are written to SYSOU\textit{nnn}. These messages include storage usage statistics and EXCP counts. NGT Recover dynamically allocates a SYSOU\textit{nnn} DD statement automatically if one is not provided.

SORTDYN

Use the SORTDYN option to control and override installation defaults for sort work dynamic allocation.

SORTDYN YES

SORTDYN YES specifies that dynamic allocation can occur and should use the NGT Recover specification for SORTDEVT and SORTNUM.

SORTDYN NO

SORTDYN NO specifies that BMCSORT defaults for sort work dynamic allocation should be used.

RESINV \texttt{\textit{integerK}}

This option overrides the RESINV installation option ("RESINV=0K" on page 591) and specifies the amount of memory that BMCSORT reserves below the 16-MB line to allow for IDCAMS processing. The installation default value of RESINV=0K allows BMCSORT defaults to be used.

\textbf{Note}

BMC recommends that you use the default value so that BMCSORT uses its algorithms effectively.

For NGT Recover versions earlier than version 4.1.00, which do not use BMCSORT, the following information may apply:

- If the system sort routine is PLSORT, always set RESINV=0K.
- RESINV corresponds to the SyncSort RSRVTI installation option.
- For DFSORT, RESINV is used only when DFSORT is program-invoked and the DFSORT parameter SIZE=MAX is in effect.
SMCORE *(integerK, integerK)*

This option overrides the SMCORE installation option ("SMCORE=(0K,0K)" on page 591) and specifies the maximum amount of memory that BMCSORT uses. The first value specifies the total amount of memory used both above and below the 16-MB line. The second value specifies the amount of memory used below the 16-MB line. The default values of SMCORE=(0K,0K) allow BMCSORT defaults to be used.

**Note**

BMC recommends that you use the SMCORE=(0K,0K) defaults so that BMCSORT uses its algorithms effectively.

For NGT Recover versions earlier than version 4.1.00, which do not use BMCSORT, the following information may apply:

- For SyncSort, both the first and the second values are used because SyncSort does use memory both above and below the 16 MB line.
- For DFSORT, only the first value is used because DFSORT does not allow separate specification of memory usage below the 16 MB line.

For more information, see "Managing sort performance" on page 474 if your system sort routine is DFSORT, SyncSort, or another package.

**MAXDRIVES integer**

The MAXDRIVES option controls the number of tape drives that are used during recovery. If a value for MAXDRIVES is not coded, the limit specified by the MAXDRIVE installation option is used.

For NOLOGSORT runs, NGT Recover arranges the process to use no more than the MAXDRIVES value. For LOGSORT runs, if more drives are required for the MERGE processing than the MAXDRIVES value, NGT Recover does not start the process. For LOGSORT runs, if MAXDRIVES is less than MAXLOGS, no more than the number of MAXDRIVES tape log data sets are opened at one time.

**MAXLOGS integer**

The MAXLOGS option is part of the concurrent log file reading feature. Use this option to perform the following tasks:

- Control the amount of memory that is used during the recovery
- Reduce the contention that is caused by reading many log files in parallel

MAXLOGS must be greater than 0.
When MAXLOGS is specified, NGT Recover will not allocate and read more log files concurrently than indicated by this option. If MAXLOGS is not coded on the OPTIONS command statement, the default is the value of the MAXLOGS installation option.

In addition to the global control that is provided by MAXDRIVES, use MAXLOGS to further control the number of tape drives that are used for the log files. If the value of MAXLOGS is less than MAXDRIVES, the number of tape drives that are used for log files is limited to the value of MAXLOGS.

**OUTCOPY**

OUTCOPY specifies how image copies are created for partitioned objects. The presence of OUTCOPY on the OPTIONS command statement overrides, for this recovery, the setting of OUTCOPY in the installation options.

Output copies are produced in the SYSTEMPAGES YES format when possible.

**OUTCOPY ASCODED**

OUTCOPY ASCODED designates that image copies are created as specified for the RECOVER TABLESPACE, RECOVER INDEX, or RECOVER INDEXSPACE command statement. When DSNUM ALL is specified, an image copy is created for the object as a whole. If the recover specification is for an individual data set (DSNUM n), a copy is made of that specific data set.

**OUTCOPY BYPART**

OUTCOPY BYPART designates that each partition of a partitioned object is copied to a separate image copy.

**RICHK**

RICHK specifies if NGT Recover will check for referential integrity constraints on the objects being recovered.

**RICHK YES**

RICHK YES checks for referential integrity constraints and sets check pending (CHKP) status for dependent objects not recovered in the same step to the same PIT.

**RICHK NO**

RICHK NO does not check for referential integrity. NO is the default value.
AUX

The AUX option specifies if auxiliary objects will be included with the recovery of the base table spaces.

The AUX option is available on the OPTIONS command or in the table space specification for any of the following NGT Recover commands:

- RECOVER TABLESPACE
- RECOVER INDEX(ALL) TABLESPACE
- RECOVER INDEXSPACE(ALL) TABLESPACE
- REBUILD INDEX(ALL) TABLESPACE
- LOGSCAN TABLESPACE
- LOGSCAN INDEX(ALL) TABLESPACE
- LOGSCAN INDEXSPACE(ALL) TABLESPACE
- MIGRATE

If you do not specify AUX on one of these commands, NGT Recover uses the value of the AUX installation option (“AUX=NO” on page 578), which has a default value of NO.

**Note**

NGT Recover ignores the AUX option when you specify OBJECTSET. Also, the AUX option has no effect for the ACCUM TABLESPACE, RECOVER UNLOADKEYS, and RECOVER BUILDINDEX commands.

When you specify AUX ALL, AUX XML, or AUX LOB, you cannot specify the following options:

- TOCOPY dataSetName
- OBIDXLAT without defaults
- OUTCOPYDSN
- RECOVERYDSN

Valid values are NO, ALL, XML, and LOB.

**AUX NO**

NGT Recover does not include XML or LOB objects. You may include HISTORY spaces with the HISTORY keyword when a table space with system-maintained temporal tables is specified in SYSIN. For more information, see “HISTORY” on page 118.
You can include ARCHIVE spaces with the ARCHIVE keyword when a table space with an archive table is specified in SYSIN. For more information, see ARCHIVE on page 119.

**AUX ALL**

NGT Recover includes XML and LOB objects for the base table space. It does not include HISTORY spaces with their related auxiliary objects when a table space with system-maintained temporal tables is specified in SYSIN. You may include HISTORY objects with the HISTORY keyword. See “HISTORY” on page 118.

You can include ARCHIVE spaces with the ARCHIVE keyword when a table space with an archive table is specified in SYSIN. For more information, see ARCHIVE on page 119.

**AUX XML**

NGT Recover includes all XML data objects along with the XML base table space. If the table space specification is found on a RECOVER TABLESPACE or SIMULATE TABLESPACE statement, all implicitly created XML table spaces will be processed along with the base XML table space. If the table space specification is found on a RECOVER INDEX(ALL), REBUILD INDEX(ALL), or SIMULATE INDEX(ALL) statement, all implicitly created XML and node ID indexes on the XML base table space as well as all explicitly created non-LOB indexes will be processed.

**AUX LOB**

NGT Recover includes all LOB data objects along with the LOB base table space. If the table space specification is found on a RECOVER TABLESPACE or SIMULATE TABLESPACE statement, all table spaces created with the LOB attribute will be processed along with the base LOB table space. If the table space specification is found on a RECOVER INDEX(ALL), REBUILD INDEX(ALL) or SIMULATE INDEX(ALL) statement, all indexes on the LOB base table space and all non-LOB explicitly created indexes will be processed.

*Note*

Specifying AUX … on the OPTIONS statement in SYSIN applies AUX … to all RECOVER statements. To override AUX in the OPTIONS statement, specify AUX … only on the RECOVER statements you wish to override.

**HISTORY**

When you specify HISTORY, NGT Recover includes history spaces when a table space with system-maintained temporal tables is specified in SYSIN.

You must specify AUX to use the HISTORY keyword. The meaning of the HISTORY keyword depends on the AUX option specified, as follows:
AUX NO HISTORY includes regular tables, system-time base tables, and their related HISTORY tables. It does not include related XML or LOB objects.

AUX ALL HISTORY includes regular tables, system-time base tables and their related HISTORY tables, and all related XML or LOB objects.

AUX XML HISTORY includes regular tables, system-time base tables and their related HISTORY tables, and all related XML objects.

AUX LOB HISTORY includes regular tables, system-time base tables and their related HISTORY tables, and all related LOB objects.

**ARCHIVE**

When you specify ARCHIVE, NGT Recover recovers archive-enabled base tables and their related archive tables.

You must specify the AUX option with ARCHIVE to tell NGT Recover whether to include auxiliary XML or LOB objects in the recovery:

- **AUX NO ARCHIVE** (or **AUX ARCHIVE**). It does not include related XML or LOB objects.
- **AUX ALL ARCHIVE** includes all related XML and LOB objects.
- **AUX XML ARCHIVE** includes all related XML objects.
- **AUX LOB ARCHIVE** includes all related LOB objects.

**USEHDROBIDS**

Use the USEHDROBIDS option to indicate if the OBIDs in the header are valid or not.

When you specify OBIDXLAT (*OBIDXLAT specification on page 191*) with INCOPY FULL SNAPSHOT TOCOPY syntax and do not specify OBIDs, NGT Recover looks at the OBIDs in the header page to determine if the OBIDs for the source and target are the same. If the OBIDs are the same, NGT Recover does not use a MERGE phase to update the OBIDs. Skipping the MERGE phase makes migration faster because it eliminates a read and write of every page of the space. (NGT Recover may still do an "optimized merge" phase to just update the header page.)

Sometimes the OBIDs in the header page are not correct. An example of when this situation might occur is when the space is copied from another object and the OBIDs in the header page are not translated correctly. If the OBIDs in the header page are not correct, NGT Recover might skip the MERGE phase when it should not, or the MERGE phase might fail because NGT Recover detects that the OBIDs in the header page do not match the OBIDs in the rest of the data. Under circumstances like these,
you can use USEHDROBIDS to tell NGT Recover if the OBIDs in the header are valid or not.

If you do not code USEHDROBIDS on the OPTION command statement, the default is the value of the USEHDROBIDS installation option ("USEHDROBIDS=YES" on page 596), which defaults to YES.

**USEHDROBIDS YES**

When you specify USEHDROBIDS YES, NGT Recover uses the OBIDs in the header page and skips the MERGE phase if possible. If NGT Recover does perform the MERGE phase and detects that the OBIDs are not correct, NGT Recover ends with an error message.

**USEHDROBIDS NO**

When you specify USEHDROBIDS NO, NGT Recover always does a MERGE phase and ignores the OBIDs in the header page.

**RECOVERSCOPE**

The RECOVERSCOPE option allows you to specify a SCOPE option that should be applied to all RECOVER command statements in the SYSIN.

The RECOVERSCOPE option settings are UPDATED, ALL, PENDING, and STATUS(status1,status2,…). For more information about these settings, see “SCOPE specification” on page 200.

If you do not specify a RECOVERSCOPE setting on the OPTION command, the default RECOVERSCOPE settings are as follows:

- SCOPE UPDATED is the default option for all local point-in-time forward recoveries. However, if any of the following conditions is met, the default remains SCOPE ALL:
  - IXRECP YES is specified in the OPTIONS command or in the default options
  - RICHK YES is specified in the OPTIONS command or in the default options
  - NOSYSLGRNG is specified in the OPTIONS command
  - OUTCOPY ONLY is in use
  - OUTCOPY YES is requested when OPTIONS OUTCOPY BYPART is not in effect

- SCOPE ALL is the default for all other recoveries.
REBUILDSCOPE

The REBUILDSCOPE option allows you to specify a SCOPE option for all REBUILD command statements in the SYSIN.

If you do not specify a REBUILDSCOPE setting on the OPTIONS command, SCOPE ALL is the default option for all REBUILD requests.

The REBUILDSCOPE option settings are ALL, PENDING, and STATUS (status1, status2,…). For more information about these settings, see “SCOPE specification” on page 222.

DEFINE

DEFINE specifies whether DEFINE NO objects will be instantiated.

**Note**
IBM DB2 Version 11 or later supports the DEFINE option.

DEFINE YES

NGT Recover instantiates DEFINE NO objects, if necessary, and updates the catalog and DBD to reflect that the object has been defined.

DEFINE NO

NGT Recover does not process DEFINE NO objects. NO is the default value.

RESOURCE SELECTION

Use the RESOURCE SELECTION option to indicate a preferred order in the selection of image copies, logs, and change accumulation files.

RESOURCE SELECTION LOGS

You can specify RESOURCE SELECTION as ACT1, ACT2, ARC1, and ARC2 in any order. ACT1 and ACT2 indicate the primary and dual (secondary) active logs. ARC1 and ARC2 indicate the primary and dual (secondary) archive logs. The default order is ACT1, ACT2, ARC1, ARC2. You can omit references to log copies that you do not want considered. However, if you omit references to active logs, they may be used anyway if the required log ranges are not yet archived.

You use RESOURCE SELECTION LOGS on the OPTION command to override the USELOGS installation option (see “USELOGS=(ACT1,ACT2,ARC1,ARC2)” on page 597).
**RESOURCE SELECTION ACCUMS**

You can specify RESOURCE SELECTION as LP, LB, RP, and RB in any order. LP and LB indicate the primary and secondary local change accumulation files. RP and RB indicate the primary and secondary remote change accumulation files. The default order is LP, LB when operating as a local site and RP, RB when operating in a recovery site. You can omit references to copies of the resource that you do not want considered.

**RESOURCE SELECTION COPIES**

You can specify RESOURCE SELECTION as LP, LB, RP, RB, FC, and SB in any order. LP and LB indicate the primary and secondary local image copies. RP and RB indicate the primary and secondary remote image copies. FC indicates an IBM FlashCopy.

SB indicates system backup copies. Similar to SYSTEMPAGES NO copies, if the space is versioned or compressed, any indexes that need to be rebuilt must be rebuilt in a separate step after the table space is recovered. Index recoveries can be done concurrently with table space recoveries. NGT Recover can use only system backups that are on disk.

The default order is FC, LP, LB when you are operating as a local site, and RP, RB, FC when you are operating in a recovery site. SB is not included in default processing. You can omit references to copies of the resource that you do not want NGT Recover to consider.

**BACKOUT**

The BACKOUT option invokes the backout strategy for point-in-time recovery by using log points (TORBA, TOLOGPOINT). This strategy assumes that spaces are undamaged and that you require a reset to a specific point in time. The spaces and the log records between the point in time and the current point are used to back out to the required state. LOGSORT is required to properly order the log records.

*Note*

The space must not be in RECP, RECP*, RBDP, RBDP*, PSRCP, PSRBD, GRECP, WEPR, or STOPE status or have an LPL range.

BACKOUT also cannot be used for LOB spaces or for spaces having a logging attribute of NOT LOGGED.

If you do not specify BACKOUT on the OPTIONS command, NGT Recover uses the value of the BACKOUT installation option ("BACKOUT=AUTO" on page 580), which has a default value of AUTO. If you are not using a Recovery Management password, point-in-time recoveries use a standard forward recovery.
The BACKOUT option causes all point-in-time recoveries specified in the SYSIN to use the backout strategy. If the BACKOUT option is specified on the OPTIONS command statement and all of the RECOVER command statements in the SYSIN are for a recovery to the current point in time, the BACKOUT option is ignored.

If you specify the BACKOUT option on the OPTIONS command statement and the SYSIN contains at least one object being recovered to a specific point in time, all objects being recovered in the SYSIN must be recovered to a point in time. If none of the objects are being recovered to a point in time, BACKOUT on the OPTIONS command statement is ignored. For a detailed discussion of the backout recovery strategy, see “Strategies for point-in-time recovery” on page 466.

**Note**

If a table space is recovered to a specific point in time by using BACKOUT and an associated index is recovered without specifying a TOLOGPOINT, TORBA, or TOCOPY keyword in the same SYSIN, the index is recovered with BACKOUT to the same point in time as the table space.

**BACKOUT AUTO**

BACKOUT AUTO supports the BMC Recovery Management for DB2 backout to forward recovery feature.

The BACKOUT AUTO option invokes the recovery strategy in which all of the PIT recovery requests are first executed using the BACKOUT option. If any BACKOUT request fails, forward recovery is performed for the objects that were not backed out. This automation produces the fastest possible recovery with minimal intervention and is only available using the Recovery Management for DB2 solution.

**Note**

- The BACKOUT AUTO option requires a Recovery Management solution password.
- Generate BACKOUT AUTO syntax by using the RECOVERY MANAGER for DB2 component of the Recovery Management solution.

For more information BACKOUT AUTO recovery, see the *Recovery Management for DB2 User Guide*.

**BACKOUT NO**

BACKOUT NO specifies that you want to perform standard forward point-in-time recoveries.
**BACKOUT YES**

BACKOUT YES invokes the backout strategy for point-in-time recovery by using log points (TORBA, TOLOGPOINT). This strategy assumes that spaces are undamaged and that you require a reset to a specific point in time. The spaces are used with the log records between the point in time and the current point to back out to the required state. Using BACKOUT may enhance the performance of a point-in-time recovery significantly. However, if the backout recovery fails, NGT Recover stops with an error. (As opposed to BACKOUT AUTO, which will move to forward recovery if the backout recovery fails.) For more information, see “Using BACKOUT” on page 466.

The following restrictions apply:

- The space must be current as of the last logged activity and not damaged in any way. Multiple-data set, nonpartitioned spaces must have all data sets scanned (DSNUM ALL).

- No LOAD or REORG events can exist between the log point specified and the current log points. For indexes, no REBUILD INDEX events can exist in this range.

- No prior point-in-time recovery with a START_RBA greater than the log point requested and a PIT_RBA less than the log point requested can exist.

- BACKOUT may not be requested with LOGONLY or LOGAPPLY ONLY.

---

**Note**

BACKOUT uses only logs and spaces but requires that the spaces be current. The LOGONLY and LOGAPPLY ONLY options imply scanning log records going forward by using a space restored to a previous state.

- Change accumulation files are not allowed with BACKOUT because they are not properly ordered. Output accumulation files are also not supported because they are defined from the point of the last image copy to the current point.

**INDEXLOG**

The INDEXLOG option setting determines the behavior of the RECOVER INDEX command. If it is not specified on the OPTIONS command statement, the setting of INDEXLOG in the installation options is used.

You might consider using INDEXLOG YES, even though you may have to change some RECOVER INDEX commands to REBUILD INDEX commands. However, if you have no immediate plans to start making index image copies, you may prefer INDEXLOG NO so that you will not have to change syntax.
The RECOVER INDEXSPACE command always invokes a recovery that is based on copies and logs, regardless of the INDEXLOG setting. REBUILD INDEX always rebuilds indexes, regardless of the INDEXLOG setting.

**INDEXLOG YES**

INDEXLOG YES causes the RECOVER INDEX command to use copies, log, or both for index recovery.

**INDEXLOG NO**

INDEXLOG NO causes the RECOVER INDEX command to rebuild the index by extracting the keys from the table space. When you specify INDEXLOG NO, the RECOVER INDEX and the REBUILD INDEX commands are synonyms.

**Note**

In this book, references to REBUILD INDEX may be replaced with RECOVER INDEX if INDEXLOG NO is specified.

**INDEXLOG AUTO**

INDEXLOG AUTO causes the RECOVER INDEX command to recover the index from image copies and log if possible. However, if NGT Recover determines that it cannot recover the index (for example, because of missing image copy or a point-in-time recovery), the RECOVER INDEX request automatically converts to a REBUILD INDEX request. INDEXLOG AUTO supports BACKOUT recoveries.

Conversion from RECOVER INDEX to REBUILD INDEX is not supported for the following options:

- **OUTCOPY**: The OUTCOPY option is not supported for REBUILD INDEX so this request is not eligible for conversion.

- **OBIDXLAT**: The OBIDXLAT option is not supported for REBUILD INDEX so this request is not eligible for conversion.

- **TOLOGPOINT**: If the TOLOGPOINT option is included in the RECOVER INDEX request, the request is only eligible for conversion to REBUILD INDEX if the associated table space is also recovered to the same log point in the same job step.

- **TOCOPY**: If the TOCOPY option was included in the RECOVER INDEX request, the request is only eligible for conversion to REBUILD INDEX if the associated table space is also recovered in the same job step.
- DSNUM: If the RECOVER INDEX command was for a specific data set of a nonpartitioned index, the request is not eligible for conversion to REBUILD INDEX.

**IXRECP**

When you are performing a point-in-time recovery, the IXRECP option setting determines whether indexes that are not rebuilt or recovered in the same run as their associated table spaces are set to RECP or RBDP status.

Use the IXRECP option when a point-in-time recovery of a table space is performed. In that type of recovery, it is important that any indexes on the space are rebuilt or recovered so that index data is synchronized with the data to which it refers. You can use IXRECP to have the pending status set for the indexes that are associated with the recovered table spaces, which forces index recoveries before the data can be accessed.

If IXRECP is not present in the OPTIONS command statement, the setting of IXRECP is determined by the IXRECP installation option.

When you are running Recovery Management and using the TRANSFORM option, you need the IXRECP option because any table space that is transformed must either have the index transformed or rebuilt. The RBDP status will make it obvious if a target index does not reflect the changes made to the table space.

**IXRECP YES**

IXRECP YES tells NGT Recover to issue a warning message for each index that has not been recovered with the table space and to put each such index in RECP or RBDP status. A later index recovery or rebuild removes the RECP or RBDP status. If indexes are rebuilt or recovered in the same run as the table space, they are not placed in RECP or RBDP status.

**IXRECP NO**

IXRECP NO, the default, causes NGT Recover not to set the RECP or RBDP status for indexes that are associated with a table space that is recovered to a point in time. No message is issued regarding the indexes.

**TRTCH**

The TRTCH option allows the use of Improved Data Recording Capability (IDRC) tape compression when creating output change accumulation files.
TRTCH NONE

This value is the default and specifies the use of the system default for the allocation of output change accumulation files. TRTCH NONE is equivalent to not specifying TRTCH.

TRTCH COMP

TRTCH COMP specifies that IDRC compression is to be used for output change accumulation files.

TRTCH NOCOMP

TRTCH NOCOMP specifies no IDRC compression for output change accumulation files (even if the system default is TRTCH COMP).

XBMID ssid or XBMID xbmGroup

The XBMID option specifies the EXTENDED BUFFER MANAGER (XBM) subsystem ID (ssid) or XBM group name (xbmGroup) to use when you restore Instant Snapshot copies or when you use the zIIP redirection capability.

ssid is the unique identifier that you specified when you installed XBM. If you are using XBM in a DB2 data sharing environment, you can use the xbmGroup name in place of ssid. The xbmGroup value is the name of the XBM coupling facility group defined to the XBM subsystem.

For the zIIP redirection capability, if you specify an XBM subsystem and ZIIP ENABLED (“ZIIP” on page 128 and “ZIIP=ENABLED” on page 599) is in effect, NGT Recover attempts to use that subsystem to enable zIIP processing. If that subsystem is not available or if it is not at the correct maintenance level, zIIP processing is not enabled.

If you do not specify an XBM subsystem either with XBMID on the OPTION command or with the XBMID installation option, NGT Recover searches for an XBM subsystem at the appropriate maintenance level to enable zIIP processing.

For recovery using Instant Snapshots, NGT Recover does not discover the XBM subsystem. Use the XBMID option in either the installation options module or on the OPTION statement to specify it. Instant Snapshots are made by NGT Copy with XBM and are registered in BMCXCOPY. For more information, see the BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual.

You use XBMID on the OPTION command to override the XBMID installation option (see “XBMID=ssid or xbmGroup” on page 598).
For specific information about valid characters for XBMID and the pattern matching capabilities of XBM, Instant Snapshots, and zIIP redirection, see the *EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide*.

**ZIIP**

The ZIIP option tells NGT Recover whether to attempt to use IBM System z Integrated Information Processors (zIIPs). NGT Recover can use enclave service request blocks (SRBs) to enable zIIP processing automatically while running jobs. Using zIIP processing can reduce the overall CPU time for NGT Recover jobs.

You can specify the default for the ZIIP command option in your installation options module by using the ZIIP installation option (“ZIIP=ENABLED” on page 599). NGT Recover was shipped with a default value of ENABLED for this option. The ZIIP option on the OPTION command overrides the default that is in the installation options module.

**ZIIP ENABLED**

ZIIP ENABLED tells NGT Recover to attempt to offload eligible processing to an available zIIP. If the zIIP is busy or not available, normal processing continues on a general-purpose processor.

To enable and use zIIP processing with NGT Recover, you must:

- Have an installed authorized version of XBM or SUF
- Start and maintain an XBM subsystem in your environment
- Have a zIIP available in your environment

**Tip**

When you use zIIP processing, BMC recommends that you specify IIPHONORPRIORITY=YES in the IEAOPT member of SYS1.PARMLIB.

You can specify a particular XBM subsystem to use by specifying a value for the XBMID installation option (“XBMID=ssid or xbmGroup” on page 598) or XBMID option on the OPTION command (“XBMID ssid or XBMID xbmGroup” on page 127).

XBM and SUF are licensed, installed, and maintained separately from NGT Recover. You can use either XBM or SUF, depending on the license that you have obtained:

- A license for the full version of the XBM product authorizes you to use all features of XBM.
- A license for SUF authorizes you to use only the snapshot and zIIP-processing features of XBM.
For more information about XBM and SUF, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

**ZIIP DISABLED**

ZIIP DISABLED tells NGT Recover to not attempt to use zIIP processing.

**DATAMVR programName**

Use the DATAMVR option to override the DATAMVR installation option value. The DATAMVR installation option provides XBM with the name of the program to use to copy a data set if a data set snap fails. To use DFDSS as the data mover, specify DATAMVR=ADRDSSU.

**KEYSORT**

Use the KEYSORT option to override the SORTDEVT and SORTNUM values in the installation options module for all REBUILD INDEX or RECOVER UNLOADKEYS, or RECOVERBUILDINDEX index sorts within a job step. The values that are specified in the OPTIONS command with KEYSORT can be overridden by SORTDEVT and SORTNUM on the REBUILD INDEX or RECOVER UNLOADKEYS, or RECOVERBUILDINDEX command statements.

When performing a REBUILD INDEX or a RECOVER UNLOADKEYS or RECOVERBUILDINDEX job, NGT Recover invokes BMCSORT, which allocates the required temporary sort work data sets. You can specify one of the following courses of action:

- Specify SORTNUM and SORTDEVT to direct the allocation
- Specify neither SORTNUM nor SORTDEVT and let BMCSORT allocate the work data sets according to sort rules
- Specify neither SORTNUM nor SORTDEVT and specify SxxxWKnn DD statements in the JCL
- Specify only SORTDEVT and let BMCSORT determine the number of data sets

If you use SxxxWKnn DD statements in the JCL, any SORTDEVT and SORTNUM specifications that are present are ignored. If you do not use SxxxWKnn DD statements and do not specify SORTDEVT or SORTNUM, the values that are supplied in the NGT Recover installation options module are used.
**SORTDEV** deviceType

SORTDEV deviceType specifies the device type for the temporary sort work data sets that BMCSORT uses.

If you specify SORTDYN NO (“SORTDYN” on page 114), BMCSORT defaults are used even if you specify a value for SORTDEV. Otherwise, NGT Recover uses its internal default value, SYSDA, if you do not specify SORTDEV.

**SORTNUM integer**

The SORTNUM option affects the allocation of sort work files when BMCSORT is allocating your sort work files dynamically. You can specify an integer value of 1 through 255.

When you specify this option, BMCSORT dynamically allocates the number of sort work files that it needs for each sort task up to the maximum that is illustrated in the following formula:

\[
\text{maximum dynamically allocated sort work files} = n - \text{preallocated sort work files}
\]

If you specify integer from 1 through 32, \(n\) equals 32. If you specify integer greater than 32, \(n\) equals integer.

**Note**

Preallocated sort work files include sort work files that are allocated in your JCL.

If you do not specify a value for SORTNUM, NGT Recover uses the installation option default.

**ANALYZE**

The ANALYZE option prints a recovery plan before executing that plan.

The following information about objects is included in the plan:

- Names of any image copy data sets on which activities are or will be based
- Names of any log data sets on which activities are or will be based
- Names of any change accumulation files on which activities are or will be based
- Log ranges, if any, on which activities are or will be based
- Number of log pages to be read
- Record sizes for index sort work data sets
- Phases to occur during execution
- Steps to occur within each phase
You can use this information to allocate space more accurately for log sort work data sets, index sort work data sets, and index key work data sets, and limit abends that are caused by inadequate data set allocations. You can also use these statistics with historical information about the times that are required to perform the various operations to estimate recovery time.

**Note**
The value that is specified for ANALYZE is effective for all RECOVER and REBUILD commands that follow. You can code the ANALYZE option on individual commands, but the value that you code must match the value that is coded on the OPTIONS command. The LOGSCAN command ignores the ANALYZE option.

**ANALYZE YES**

ANALYZE YES is the default and provides information about the objects in the recovery plan.

**ANALYZE NO**

If you specify ANALYZE NO, NGT Recover does not provide any recovery plan information, but does provide object summaries.

**ANALYZE ONLY**

If you specify ANALYZE ONLY, NGT Recover provides the same information as it provides with ANALYZE YES and then stops the job when the ANALYZE phase is complete. You can use the data from the ANALYZE phase to determine the resources required and what will happen during your NGT Recover run. You cannot restart a NGT Recover job that specifies ANALYZE ONLY, but you can start a new recovery job.

Use the ANALYZE ONLY option to determine which phases will occur with a specific recovery request, or which copy data sets and log data sets will be used in the recovery.

If you provide a SYSPICK DD statement, NGT Recover generates a list of all of the input tape and cartridge volumes that are allocated during recovery. For more information about SYSPICK, see “NGT Recover data sets and NGT Recover DD statements” on page 319.

If you specify OPTIONS EARLYRECALL with ANALYZE ONLY, the recall of the data sets needed for the recovery will be initiated. You must code OPTIONS EARLYRECALL explicitly to initiate the recalls because OPTIONS NOEARLYRECALL is the default for ANALYZE ONLY executions.
Use ANALYZE ONLY on the OPTIONS command for the recovery estimation feature available in the BMC Recovery Management for DB2 solution. The estimation information generated when you use this option in Recovery Management is stored in DB2 recovery history tables. For more information, see the Recovery Management for DB2 User Guide.

**SIMULATE YES**

The SIMULATE YES option specifies that you want to run in simulation mode, which provides a way for you to validate that you can recover your application data.

You can verify that needed recovery resources are valid and available and that log apply can be done.

---

**Note**

Simulation is a feature of the BMC Recovery Management for DB2 solution and requires a valid Recovery Management solution password.

---

In simulation mode, NGT Recover proceeds with all recovery activities that are not destructive. Destructive actions that could affect real database objects or recovery environment are simulated or suppressed.

There are several limitations and requirements you should take in account when you use the SIMULATE YES option:

- The SIMULATE YES option cannot be specified with the ANALYZE ONLY, or ANALYZE SCANONLY options.
- The LOGSCAN command cannot be used if you specify the SIMULATE YES option.
- Point-in-time recovery is not allowed if the SIMULATE YES option is specified.
- The requested recovery scenario can require unloading keys from the real table space (partition) if REBUILD or UNLOADKEYS is requested for the table space (partition) not recovered in the same step. Such actions will be suppressed if the SIMULATE YES option is specified.
- Objects may be in any initial status and there is no impact on the status or contents of the real database objects if the SIMULATE YES option is specified.
- No registration into DB2 or BMC tables occurs and no restarts are possible if the SIMULATE YES option is specified.
- Some options related to the suppressed or simulated actions will have no impact if the SIMULATE YES option is specified.
In the following recovery scenarios, NGT Recover creates and uses temporary data sets (SIMDBC or SIMDBD), instead of real spaces (DSNDBC or DSNDBD), for simulation:

- Recoveries from Inline, instant snapshot, or flashcopy image copies
- Backout, LOGONLY, or LOGAPPLY ONLY recoveries

When you simulate recovery of inline, instant snapshot, or flashcopy image copies, NGT Recover snaps the image copy into temporary data sets to process the simulation.

When you simulate a backout, LOGONLY, or LOGAPPLY ONLY recovery, NGT Recover snaps the original spaces into temporary data sets to process the simulation.

Except when temporary data sets are required, any output activity for the spaces and output copies (including data set allocation, creation, and extension) is only simulated if the SIMULATE YES option is specified. No real I/O or requests to the system services take place. When temporary data sets are required, output activity is restricted to the temporary data sets.

Use SIMRCVR and SIMRBLD instead of regular RECOVER and REBUILD if the SIMULATE YES option is specified.

For more information about recovery simulation, see *Recovery Management for DB2 User Guide*. You can use RECOVERY MANAGER for DB2 and its online interface to generate JCL for the simulation mode for both application and system recovery.

### SIMULATE YES LOGPOINT X'logPoint'

Use the SIMULATE YES LOGPOINT X'logPoint' option to specify the "pseudo current" point for the simulated recovery. A request to simulate recovery to the current point is done if LOGPOINT X'logPoint' is omitted.

### WTOR

If one or more spaces in the RECOVER command statement remain in STOPP status, WTOR provides the flexibility to issue a write to the operator requesting a reply (WTOR YES) or assume an operator reply of CANCEL (WTOR NO) and terminate the job.

#### WTOR YES

NGT Recover issues a WTOR.

#### WTOR NO

NGT Recover assumes an operator reply of CANCEL and terminates the job.
ON ERROR ANY CONTINUE integer or ON ERROR CONTINUE integer

The ON ERROR ANY CONTINUE integer option enables you to specify how NGT Recover is to proceed when errors are encountered.

ANY is optional and is the default value. ANY specifies that the requested action is to be taken for any recognized severe error. This value is provided to allow selection by type of error in the future.

The integer variable can be any integer value between 0 and 2,147,483,646.

ON ERROR CONTINUE integer allows integer + 1 errors before NGT Recover terminates. If integer is 0, NGT Recover stops processing immediately when the first recognized severe error occurs.

If ON ERROR CONTINUE 0 is specified, the subtask to preallocate VSAM data sets is disabled. This could increase the execution time by several seconds for each object recovered. If BACKOUT AUTO is specified through use of the Recovery Management solution, NGT Recover does not limit the number of errors allowed. For information about the BACKOUT AUTO option, see “Using BACKOUT AUTO recovery” on page 47 or the Recovery Management for DB2 User Guide.

MAXKSORT integer

The MAXKSORT option specifies the maximum number of index key sorts and index rebuilds that can run in parallel subtasks.

Note
When you rebuild the indexes of a multi-data-set, nonpartitioned table space, the UNLOADs run serially in the main task, but the REBUILDs are multtasked.

Valid values are 1 to 999. If you do not specify a value for MAXKSORT, NGT Recover uses the value of the MAXKSORT installation option, which defaults to the following formula:

\[ \text{minimum}(2 \times \text{the number of CPUs}, 12) \]

Note
Each sort requires about 256 KB of memory below the line. Values for MAXKSORT greater than 12 are not recommended.

For a description of the MAXKSORT installation option, see “MAXKSORT” on page 587.
When the value of MAXKSORT is greater than 1, NGT Recover ignores the WORKDDN option on the REBUILD INDEX command and issues a warning message. To use the WORKDDN option, specify a value of 1 for MAXKSORT. For a description of WORKDDN, see “WORKDDN DDName” on page 227.

Using the MAXKSORT option can improve recovery performance.

The MAXKSORT value determines the level of concurrency that can be achieved for index key sorts and index rebuilds. If this value is too small, the level of concurrency could be unnecessarily limited and the size of each sort will be larger. If this value is too large, the recovery job could overuse system resources and degrade recovery performance and overall system performance.

The concurrency that is reached by using MAXKSORT is limited by the following items:

- The amount of memory that is available below the 16-MB line for BMCSORT processing
  
  In most environments, available memory below the 16-MB line creates a practical limit of 15 to 20 sorts.

- The value assigned to the KSORTSHARE option (“MAXLSORT integer” on page 136 and “KSORTSHARE=YES” on page 586)

  KSORTSHARE specifies if key sorts are shared among NGT Recover table space recoveries (MERGE phases) running in parallel.

If the key lengths of the indexes vary widely in size, MAXKSORT increases efficiency because it allocates only the amount of memory that is needed for each key.

If the index rebuild includes both partitioned and nonpartitioned indexes, MAXKSORT, if set to a value of 3 or greater, could allow the sorting of the partitioned indexes separately from the nonpartitioned indexes and might improve efficiency.

For each table space, index keys for all indexes being rebuilt are distributed over the number of sorts that you specify for this option and these sorts can run in parallel. If NGT Recover is recovering a partitioned table space and is rebuilding the partitioning index, the rebuild of each partition may be performed at the completion of the MERGE for each partition of the table space. The rebuild can run concurrently with the MERGE for the next partition if the MAXKSORT number is not exceeded. Running concurrent index key sorts and index rebuilds can increase the speed of the recovery.

Restart can cause keys that have already been extracted and sorted to be extracted and sorted again, but the restart process is relatively straightforward.
The following files are dynamically allocated if you do not code them in JCL:

- **SYSOU*nnn**: sort message files
  *nnn* is a number between 1 and the value specified for MAXKSORT.

- **SxxxWK*nnn**: key sort work files
  *xxx* is a number between 1 and the value specified for MAXKSORT. *nnn* is a number between 1 and the value specified for the SORTNUM installation option or the OPTIONS SORTNUM parameter.

When you use dynamic allocation for these files, BMCSORT determines the optimal number of files to use.

For more information about setting MAXKSORT, see “Setting the MAXKSORT option” on page 479.

**MAXLSORT integer**

The MAXLSORT option specifies how many log sorts NGT Recover can run in parallel and also determines the number of MERGE/SNAP/RESTORE phases that can run in parallel regardless of log requirements.

These phases run in subtasks and MAXLSORT sets the number of subtasks to use. Using the MAXLSORT option can improve recovery performance.

Valid values are 0 to 999. If you do not specify a value for MAXLSORT, NGT Recover uses the value of the MAXLSORT installation option (“MAXLSORT=0” on page 588). The MAXLSORT installation option has a default value of 0 (MAXLSORT=0), which allows NGT Recover to determine an appropriate value for MAXLSORT.

If you specify MAXLSORT 1, you set up your job to run as it would in NGT Recover version 8.1.00 and earlier and turn off parallel log sorts and parallel MERGE/SNAP/RESTORE phases. These phases then run serially in the main task in the order in which they are scheduled by the NGT Recover planning component.

The following files are dynamically allocated if you do not code them in JCL:

- **LOGOU*nnn**: sort message files
  *nnn* is the number of the log sort and is a number between 1 and the (non-zero) value that is specified for MAXLSORT.

- **LxxxWK*nnn**: sort work files
  *xxx* is the number of the log sort and is a number between 1 and the (non-zero) value that is specified for MAXLSORT. *nnn* is the number of the work data set.
example, if MAXLSORT=3 and two sort work files are required for each sort, the
DDs would be specified as follows:

L001WK01 DD...
L001WK02 DD...
L002WK01 DD...
L002WK02 DD...
L003WK01 DD...
L003WK02 DD...

When you use dynamic allocation for these files, NGT Recover determines the
optimal number of files to use.

When MAXLSORT is greater than 1, the MERGE/SNAP/RESTORE phases run in
subtasks in parallel. The order of execution of phases may be different from the
execution plan in AFRPLAN. NGT Recover prints the output for each phase for each
object in an AFRPRnnn data set ("DD statements common to all NGT Recover
executions" on page 319). NGT Recover names the files AFRPR001, AFRPR002,
AFRPR003, and so on. The maximum number of AFRPRnnn files is calculated by the
following formula:

\[
\text{value of MAXKSORT + value MAXLSORT} + 1 = \text{maximum number of AFRPRnnn files}
\]

AFRPRINT contains a list of the phases executed for each object and the name of the
AFRPRnnn data set where NGT Recover put the output for the phase.

AFRSUMRY messages are printed in execution plan order.

For more information about setting MAXLSORT and how the parallel merge phase
works, see “Setting the MAXLSORT and KSORTSHARE options” on page 481.

**KSORTSHARE**

The KSORTSHARE option specifies if key sorts are shared among NGT Recover
table space recoveries (MERGE phases) running in parallel.

If you do not specify a value on the OPTIONS command for KSORTSHARE, NGT
Recover uses the value of the KSORTSHARE installation option
("KSORTSHARE=YES" on page 586), which defaults to YES.

When you specify YES, NGT Recover uses up to the value specified for MAXKSORT
for active key sorts at any given time. If sufficient key sorts are not available when a
table space recovery begins execution, NGT Recover obtains keys later in an
UNLOAD phase.

When you specify NO, each MERGE phase has its own set of key sorts and up to
MAXKSORT * MAXLSORT key sorts can be active at any given time. Since the
number of sorts that can be active in a system is fairly small - usually no more than
30 - a value of NO for this option may severely limit the number of recovery
operations that NGT Recover can perform in parallel when index rebuilds are also requested.

For more information about setting KSORTSHARE and how the parallel merge phase works, see “Setting the MAXLSORT and KSORTSHARE options” on page 481.

**AUTOSIZE**

This option turns dynamic sizing for output image copies or change accumulation output files on or off. Valid values are YES are NO. If you specify a value for AUTOSIZE, you override the AUTOSIZE installation option, which defaults to YES.

**AUTOSIZE YES**

AUTOSIZE YES specifies that dynamic sizing of output image copies or change accumulation output files to DASD occurs.

Note

Use AUTOSIZE=YES option when you want to use dynamic space calculation of OUTCOPY files generated by NGT Recover. For more information, see “Automatic sizing of output copies dynamically allocated on DASD” on page 499.

**AUTOSIZE NO**

AUTOSIZE NO specifies that output image copies or change accumulation output files are allocated to DASD using the primary and secondary quantities that are specified in the R+/CHANGE ACCUM repository.

**RECOVERYPOINT**

The RECOVERYPOINT option supports the BMC Recovery Management for DB2 timestamp recovery feature.

Note

To use a timestamp recovery, you must have the Recovery Management solution, with its valid solution password, installed. Without the solution password, jobs that use RECOVERYPOINT fail.

BMC Recovery Management for DB2 can perform a consistent, point-in-time recovery to a user-specified timestamp by using Inflight Resolution technology. NGT Recover translates the timestamp to an RBA or LRSN, recovers the objects, then resolves all in-flight units of work for both data sharing and non-data-sharing systems. You can perform timestamp recovery either by using the online interface of RECOVERY MANAGER or by submitting a batch NGT Recover job with the
RECOVERYPOINT option. (The online interface of RECOVERY MANAGER for DB2 supports this feature on data sharing systems.)

The Inflight Resolution feature of the BMC Recovery Management for DB2 solution enables you to perform a consistent recovery to any LRSN, RBA, or timestamp. The ability to resolve inflight units of work at any point in time completely eliminates the need to perform quiesces to establish consistent recovery points during application execution. The ability to avoid quiesces can dramatically improve the availability of your DB2 data.

For more information, see the Recovery Management for DB2 User Guide and the RECOVERY MANAGER for DB2 User Guide.

The RECOVERYPOINT option specifies the recovery point and directs the utility to resolve inflight units of recovery. You specify the RECOVER statements as if a recovery to the current point in time is specified. You can have multiple RECOVER statements in the input. The RECOVERYPOINT option effectively determines what the current recovery point is. You can specify the TIMESTAMP option or the LOGPOINT option (but not both) with RECOVERYPOINT.

In the following examples, the first example shows the syntax with the TIMESTAMP option and the second example shows the syntax with the LOGPOINT option:

```
OPTION RECOVERYPOINT URIDDN(output) TIMESTAMP yyyy-mm-dd-h.mm.ss.tttttt
OUTPUT(output) UNIT SYSDA
DSNAME dataSetName
RECOVER TABLESPACE databaseName.tableSpaceName
TABLESPACE databaseName2.tableSpaceName2

OPTION RECOVERYPOINT URIDDN(output) LOGPOINT X'logPoint'
OUTPUT(output) UNIT SYSDA
DSNAME dataSetName
RECOVER TABLESPACE databaseName.tableSpaceName
TABLESPACE databaseName2.tableSpaceName2
```

You should consider the following items when you use the RECOVERYPOINT option:

- RECOVERYPOINT is not valid with the use of the ACCUM or IMPORT commands in the same SYSIN.

- RECOVERYPOINT interacts with the SIMULATION option in the following ways:

  - If OPTION SIMULATE specifies no LOGPOINT, OPTION RECOVERYPOINT determines the ending point of the simulation.

  - If OPTION RECOVERYPOINT specifies a TIMESTAMP, OPTION SIMULATE must not specify a LOGPOINT.

  - If both OPTION RECOVERYPOINT and OPTION SIMULATE specify LOGPOINT, they must both specify the same value.
To use OUTCOPY ONLY and OBIDXLAT with OPTION RECOVERYPOINT, you will need to rebuild any indexes on the target table spaces.

If the RECOVERYPOINT specifies a point on the log during a SHRLEVEL CHANGE image copy, forward recovery from that copy fails if the copy includes any changes that were inflight at the recovery point. If that happens, NGT Recover attempts to fall back to a previous copy.

If OPTION RECOVERYPOINT is specified, no other point-in-time specification (TORBA, TOLOGPOINT) is allowed.

**LOGPOINT LOGMARK logMarkName (logMarkGeneration)**

The LOGPOINT LOGMARK logMarkName option specifies the recovery point as a log mark that was created in Log Master.

Optionally, you can add a log mark generation number in parentheses, (logMarkGeneration). If you do not specify a log mark generation, NGT Recover uses the most recent version of the log mark. You can use one of the following ways to specify the log mark generation:

- If you specify a log mark generation as less than or equal to zero (\( \leq 0 \)), NGT Recover treats the generation as a relative generation. NGT Recover refers to the most recent log mark with the generation of zero (0). The previous generation of the log mark is referred to with (−1), and so on.

- If you specify a log mark generation as greater than zero (> 0), NGT Recover treats the generation as an absolute generation number and uses the specified version of the log mark.

**Note**

The way that you specify an absolute generation number for a log mark in NGT Recover is different than the way you specify the absolute generation number in Log Master.

When you create the log mark, you can specify if Log Master should create a quiesce point. You can also specify that Log Master set a log mark at a quiet point or a non-quiet point. If you know that the log mark is not at a quiesce point or a quiet point for the spaces or spaces that you are recovering, you should use a timestamp recovery by using OPTION RECOVERYPOINT LOGMARK syntax. If you know that the log mark is at a quiesce point or a quiet point, consider using RECOVER TOLOGPOINT LOGMARK syntax ("TORBA or TOLOGPOINT" on page 205) to perform a point-in-time recovery.
**Note**
You must determine if the log mark is at a quiesce or quiet point or not. If you are not sure that there is a quiesce or quiet point, a timestamp recovery may be best although it will take longer to look for inflight transactions.

**LOGPOINT X'logPoint'**

The LOGPOINT X'logPoint' option specifies the recovery point for the recovery.

**SNAP**

The SNAP option indicates if you want NGT Recover to read VSAM copies, even if the data set is not on a snappable disk.

If you do not specify SNAP on the OPTION command, NGT Recover uses the value of the SNAP installation option (“SNAP=HW” on page 592) as the default value. The default value of the SNAP installation option is HW.

**SNAP HW**

When you specify SNAP HW, NGT Recover uses a hardware snapshot to restore an Instant Snapshot or VSAM data set.

**SNAP VSAM**

When you specify SNAP VSAM, NGT Recover uses conventional VSAM I/O to restore a VSAM data set if it is not on a snappable disk.

To read a VSAM copy with SNAP VSAM, you specify the name of the VSAM data set in an INCOPY statement, just as you would if it was an Instant Snapshot copy, using the INCOPY FULL SNAPSHOT DSNAME dataSetName syntax.

SNAP VSAM also allows you to recover using a VSAM copy registered in BMCXCOPY or SYSCOPY if that copy is not on snappable disk.

**TIMESTAMP yyyy-mm-dd-h.mm.ss.tttttt**

The TIMESTAMP option specifies the recovery point. If you use TIMESTAMP, you cannot use LOGPOINT. TIMESTAMP requires a value to at least the second. You need to specify unspecified levels of detail with zeros. For example, a TIMESTAMP specification that only goes to the minute level needs to specify zeros for the seconds level.

**OPTION RECOVERYPOINT TIMESTAMP 2013-01-15-09.15.00**
The TIMESTAMP value is converted to a log point and adjusted for Daylight Savings Time. The Daylight Savings Time adjustment uses the TZRULE installation option. For information about this installation option, see “TZRULE=NONE” on page 594.

**URIDDDN (output)**

You use the URIDDDN parameter to identify an OUTPUT statement that is used to allocate a data set where inflight units of recovery are reported.

The URIDDDN parameter refers to an OUTPUT statement to capture a UNIT name and data set name (DSNAME). If you specify the SPACE parameters, NGT Recover also uses them. If the OUTPUT statement specifies a tape unit, NGT Recover ignores it. NGT Recover honors any SMS class specifications (DATAACLAS, MGMTCLAS, STORECLAS) on the OUTPUT statement. NGT Recover ignores any other OUTPUT statement parameters.

If the RECOVERYPOINT request does not include a URIDDDN specification, NGT Recover defaults the data set name to:

```
&SYSUID..D&DATE..T&TIME.URIDS
```

An example data set name that uses this default is RDAMSM.D120206.T171005.URIDS.

If the DSNAME parameter on the OUTPUT statement is a relative GDG name, NGT Recover resolves it to an absolute name before allocation.

If the request does not include a URIDDDN specification or if the URIDDDN does not include UNIT, NGT Recover uses the WKUNIT value defined in the installation options.

If the request does not include a URIDDDN specification or if the URIDDDN does not include SPACE parameters, NGT Recover defaults the space to TRK(1,1).

---

**Note**

NGT Recover deletes this data set at the end of a successful job.

---

**OUTPUT command**

This topic describes the syntax of the OUTPUT command and its options.
OUTPUT syntax

The following figure shows the syntax of the OUTPUT command.
Figure 7: OUTPUT syntax

- OUTPUT name DSNAMES dataSetName

- Common options
  - CATLG
    - YES
    - NO
  - DATACLAS name
  - MGMTCLAS name

- Disk options
  - MODELDCB dataSetName
  - STORCLAS name
  - UNIT
    - SYSALLDA
      - name
    - UNICTNT
      - integer
    - VOLCNT
      - integer

Figure 8: OUTPUT syntax—Common options

- Common options
  - CATLG
    - YES
    - NO
  - DATACLAS name
  - MGMTCLAS name

- Disk options
  - MODELDCB dataSetName
  - STORCLAS name
  - UNIT
    - SYSALLDA
      - name
    - UNICTNT
      - integer
    - VOLCNT
      - integer

Figure 9: OUTPUT syntax—Disk options

- Disk options
  - EATTR
    - NO
    - YES
    - NONE
  - MAXPRIM
    - integer
  - SPACE
    - (primary, secondary)
    - CYL
    - TRK
  - VOLUMES (volume1, volume2, ...)

Figure 10: OUTPUT syntax—Tape options

- Tape options
  - RETPD integer
  - EXPDT date
  - 99000
  - STACK
    - NO
    - YES
  - TRTCH
    - COMP
    - NOCOMP
    - NONE
OUTPUT command dependencies and prohibitions

When you use an OUTPUT command in NGT Recover, the following rules apply:

- The command statement must start with the OUTPUT command followed by the name of the descriptor that you want to use to dynamically allocate your output data sets.
- You must specify DSNAME.
- You can specify only options that apply to the media that you use. That is, all of the options must apply either to disk data sets or to tape data sets.
- You can have more than one OUTPUT command statement in a SYSIN data set, but each output descriptor must have a different name.
- Options that are specific to disk data sets and those specific to tape data sets are mutually exclusive: you cannot specify disk output data sets and tape output data sets in the same OUTPUT command statement. To specify disk and tape output data sets in the same SYSIN data set, you must use one OUTPUT command statement for the disk data sets and another for the tape data sets. The names of the descriptors must be different.

For more information, see “Allocating output image copy data sets dynamically” on page 491.

OUTPUT option descriptions

NGT Recover provides options for use with the OUTPUT command. The options are described in the order in which they are shown in the OUTPUT syntax diagram.

**OUTPUT name**

Use OUTPUT name to introduce a new output descriptor name. NGT Recover creates the named descriptor by using the options specified in the OUTPUT command statement. The value for name must not exceed eight characters.

**DSNAME dataSetName**

Specify DSNAME and a data set name (dataSetName) to specify a data set name or model to use.

DSNAME is not required if you use the OUTCOPYDSN keyword ("OUTCOPYDDN specification" on page 198), or the RECOVERYDSN keyword ("RECOVERYDDN specification" on page 198).
specification” on page 199), or both in the OUTCOPY specification on the RECOVER command statement.

You can construct dataSetName by using the symbolic variables in “Using symbolic variables” on page 494.

**Common options specification description**

This topic describes options that you can use for output data sets written to disk or tape.

**UNIT name**

Specify UNIT and a new tape or disk unit name when you want to override an existing default unit. The NGT Recover default is SYSALLDA.

NGT Recover can dynamically determine whether a unit is tape or disk.

**CATLG**

Specify CATLG to indicate whether to redefine the operating system catalog directive for the named descriptor.

**CATLG YES**

CATLG YES is the default and redefines the operating system catalog directive for the named descriptor.

If any SMS option (STORCLAS, DATACLAS, or MGMTCLAS) is used, NGT Recover forces CATLG YES.

**CATLG NO**

CATLG NO does not redefine the operating system catalog directive for the named descriptor.

**MODELD CB dataSetName**

Specify MODELD CB and a cataloged data set name (dataSetName) to redefine the model data control block (DCB) for the named descriptor.

To specify that no model DCB be used, use MODELD CB NONE.

The specified model data set must be allocated on a mounted direct access volume. NGT Recover copies the DCB information from the data set label.
You can construct $\text{dataSetName}$ by using any of the symbolic variables in “Using symbolic variables” on page 494.

**VOLCNT integer**

To set VOLCNT $\text{integer}$ for the named descriptor, specify the largest number of volumes that you expect NGT Recover to process when copying a single data set. For both tape and disk data sets, $\text{integer}$ must be equal to or greater than the number of volumes produced for the single largest output copy, whether or not you use stacked output. The default is 25. To use the operating system default, set VOLCNT to 0.

**Note**

If you are using SMS in your system, BMC recommends that you use VOLCNT 0.

**UNITCNT integer**

Use UNITCNT to specify the unit count used for dynamic allocation. Valid values are 0 (zero) to 59. The value 0, the default, means that the unit count is not specified for the allocation.

Specifying UNITCNT 2 for tape output allocates two tape drives. When a tape volume is at the end of tape, NGT Recover begins writing on the second drive immediately. This eliminates time spent waiting for tape rewind.

The total number of tape drives allowed in use at one time is specified by the MAXDRIVES option, either as coded in the OPTIONS command statement or from the MAXDRIVE installation option. If the value for UNITCNT is greater than one and would cause the total number of tape drives in use to exceed MAXDRIVES, NGT Recover decrements UNITCNT to prevent the allocation from exceeding that limit.

**DATACLAS name**

Specify DATACLAS $\text{name}$ when you want to provide an SMS data class name for the named descriptor. The value of $\text{name}$ must be a valid SMS data class name, not exceeding eight characters.

NGT Recover forces CATLG=YES when you specify DATACLAS.

**MGMTCLAS name**

Specify MGMTCLAS $\text{name}$ when you want to provide an SMS management class name for the named descriptor. The value of $\text{name}$ must be a valid SMS management class name, not exceeding eight characters.

NGT Recover forces CATLG=YES when you specify MGMTCLAS.
**STORCLAS name**

Specify STORCLAS name when you want to provide an SMS storage class name for the named descriptor. The value of name must be a valid SMS storage class name, not exceeding eight characters.

NGT Recover forces CATLG=YES when you specify STORCLAS.

**Disk options specification description**

This topic describes options that apply only to output data sets written to disk devices.

**SPACE (primary, secondary) allocation unit**

Specify SPACE to set the output allocation units (tracks or cylinders) for the named descriptor. Specify SPACE TRK to allocate the output in tracks. Specify SPACE CYL to allocate the output in cylinders. The default value for this option is CYL. Specify the values in parentheses, as in the following example: SPACE (200,100) CYL. If no SPACE specification is provided, the default is (5,5) CYL.

---

**Note**

You can use NGT Recover to automatically size the OUTCOPY files generated by NGT Recover. For more information, see “Automatic sizing of output copies dynamically allocated on DASD” on page 499.

**VOLUMES (volume1, volume2, .........., volumen)**

Specify VOLUMES to provide a list of default volumes for the named descriptor. The number of entries in the list must not exceed the value specified by VOLCNT for the named descriptor (see “VOLCNT integer” on page 147). If the data set is uncataloged, NGT Recover truncates the list recorded in SYSIBM.SYSCOPY to reflect the actual volumes used.

---

**WARNING**

Enough space must exist on the first specified volume to allocate the primary space required for the output data set.

**EATTR**

Use EATTR to specify whether a data set supports extended attributes or not. If you do not specify EATTR on the OUTPUT command, NGT Recover uses the value of the EATTR installation option ("EATTR=" on page 582). The EATTR installation option defaults to EATTR=, which is the equivalent of EATTR=NONE.
z/OS Version 1.11 or later supports the EATTR option. For earlier versions of z/OS, you must set EATTR=NONE.

You cannot use an image copy made to the cylinder-managed portion of an extended address volume (EAV) under z/OS Version 1.11 on z/OS Version 1.10 because z/OS Version 1.10 does not support sequential data sets in the cylinder-managed portion of an EAV.

You can also set EATTR to OPT or NO in the JCL.

When you create an OUTCOPY (OUTCOPY specification on page 193) and want to force the copy to the cylinder-managed space of an EAV, specify the SPACE keyword ("INCOPY specification" on page 182).

Valid values for EATTR are:

- NONE specifies no value for EATTR and allows the value for EATTR to be set by an SMS DATACLAS
  Using NONE allows you to have your environment set up to use extended attributes.

- OPT specifies that extended attributes are optional for the data set.
  You must set EATTR=OPT to allocate an extended format sequential data set. By using EATTR=OPT, NGT Recover supports sequential data sets in the cylinder-managed portion of EAVs.
  Extended format sequential data sets must be allocated on SMS-managed volumes and the size of the data set must be greater than the EAV break point, which is typically 10 cylinders.

- NO specifies that the data set cannot have extended attributes.

**MAXPRIM integer**

Specifying MAXPRIM allows you to set a maximum amount of disk space (in the units specified by SPACE) to allocate as primary space. Valid values are 0 through 65535. A nonzero value for integer establishes an upper limit for primary space allocation, while a value of zero specifies no limit.

**Note**

Although this option is not required for use with the AUTOSIZE=YES option ("AUTOSIZE" on page 138), you might find it useful when you want to use dynamic space calculation of OUTCOPY files generated by NGT Recover. For more information, see "Automatic sizing of output copies dynamically allocated on DASD" on page 499.
Tape options specification description

This topic describes the options that apply only to copies written to tape.

STACK

The STACK option tells NGT Recover whether to stack the output copies from multiple RECOVER executions contiguously on the same tape volumes. For information about using tape stacking, see “Stacking copies on tape” on page 498.

STACK NO

STACK NO, the default, tells NGT Recover not to stack output copies contiguously on tape.

STACK YES

STACK YES tells NGT Recover to stack output copy data sets (of the same type) from multiple RECOVER command statements contiguously on the same tape.

WARNING

If you are using Tape Mount Management (TMM), be aware that TMM intercepts any data set allocation whether dynamic or otherwise. If you want the copies on tape and use STACK YES with TMM, add the NGT Recover program AFRMAIN to the TMM exclusion list. If NGT Recover detects that the allocation has gone to disk instead of tape, it discontinues stacking.

TRTCH

The TRTCH option allows the use of Improved Data Recording Capability (IDRC) tape compression for the named descriptor.

TRTCH NONE

This value is the default and specifies the use of the system default for compression. TRTCH NONE is equivalent to not specifying TRTCH.

TRTCH COMP

TRTCH COMP specifies that IDRC compression is to be used for output files.

TRTCH NOCOMP

TRTCH NOCOMP specifies no IDRC compression for output files (even if the system default is TRTCH COMP).
RETPD integer

Specify RETPD integer to set a new retention period (in days) for the current output data set. The value of integer must be in the range of 1 through 9999.

Note
When EXPDT is specified, it takes precedence over RETPD.

EXPDT date

Specify EXPDT date to set a new expiration date for the current output data set. The default value of the EXPDT option is 99000, which indicates no expiration date.

Note
When it is specified, EXPDT takes precedence over RETPD.

The value of date must be in the format YYDDD, YYYYDDD, or YYYY/DDD, where YYYY is the 4-digit year, YY is the last two digits of the year, and DDD is the 3-digit Julian day (001 through 366).

Note
A date with a two-digit year is passed as is to dynamic allocation. For years beyond 1999, depending on your environment, this date may not produce the desired result. BMC recommends using a four-digit year.

RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands

This topic describes the syntax of the RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands and their options.

RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax

The following figures show the syntax of the RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands.
Figure 11: RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax diagram
Figure 12: RECOVER syntax—Table space specification

Table space specification

1 Not valid with TRANSFORM
2 Not valid with BACKOUT
Figure 13: RECOVER syntax—Single index specification

Single index specification

RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands

\[ INDEX \{\]
  \[ creatorID.\]
  \[ indexName \]
  \[ \}

\[ INDEXSPACE \{\]
  \[ databaseName.\]
  \[ indexSpaceName \]
  \[ DSNDB04 \]
  \[ \}

\[ DSNUM \]
  \[ ALL\]
  \[ integer\]
  \[ begin : end\]

\[ NOWORKDDN\]
\[ OBIDXLAT specification\]
\[ NOWORKDDN\]

\[ Non-registered copy or INLOG specification\]
\[ FROMRBA or FROMLOGPOINT specification\]
\[ OUTCOPY specification\]

\(^1\) Valid with INDEXLOG AUTO
Figure 14: RECOVER syntax—Multiple index specification

1 Allowed with multiple recoveries only if dynamic outcopy allocation (OUTPUT) syntax is used.
2 Valid with INDEXLOG AUTO.
Figure 15: RECOVER syntax—OBJECTSET specification

**OBJECTSET specification**

```
OBJECTSET creator.name
```

```
EXCLUDE ( dbname.
  qualifier.idx
  DSNUM ALL
  integer
  begin:end
)
```

```
OBIXLATE
  RESET
```

Figure 16: RECOVER syntax—FROMRBA or FROMLOGPOINT specification

**FROMRBA or FROMLOGPOINT specification**

```
FROMRBA
FROMLOGPOINT
```

```
LASTQUIESCE
LASTCOPY
LASTARCHQ
LASTSHUTDOWN
X'logPoint'
```
Figure 17: RECOVER syntax—INCOPY specification

INCOPY specification

DSNAME dataSetName
  MODEL dataSetName
    DSIZE (integerG)¹
    PIECESIZE (...)¹
      integerK
      integerM
      integerG
    SHRLEVEL
      REFERENCE
        CHANGE
    ENCRYPTED
    TIMESTAMP timestamp
    COMPRESSED ²
    INVOLUME
      CATALOG
        volumeSerialNumber
        INDEV deviceType
      INSEQNO integer

¹ Valid only with TRANSFORM
² Not valid with table space specification
Figure 18: RECOVER syntax—INDEPENDENT OUTSPACE specification

INDEPENDENT OUTSPACE specification

INDEP OUTSPACE

MODEL Vcat.MBCDBC.databaseName.objectName.10001.znnn

MODEL userNamedDataSet

¹ Not valid with BACKOUT

Figure 19: RECOVER syntax—LOGSORT specification

LOGSORT specification

LOGSORT

SORTDEVT deviceType

SORTNUM integer

NUMREC

CALC

NOEST

EST integer

ABS integer

AVGRECSZ integer

AVGRECSZ integer
Figure 20: RECOVER syntax—Non-registered copy or INLOG specification

Non-registered copy or INLOG specification

\[\text{INCOPY} \quad \text{FULL} \quad \text{INLINE}^1 \quad \text{SNAPSHOT} \quad \text{INLOG RBA}^{1,2} \quad \text{INLOG LOGPOINT}^{1,2} \quad \text{X'logPoint'} \quad \text{INCOPY specification} \quad \text{INCREMENTAL}^1 \quad \text{INCOPY specification} \quad \text{NOCOPYPEND}\]

1 Not valid with index specification
2 Not valid with BACKOUT
Figure 21: RECOVER syntax—OBIDXLAT specification

OBIDXLAT specification

1 Not valid with BACKOUT
Figure 22: RECOVER syntax—OUTCOPY specification

OUTCOPY specification

- OUTCOPY
- NO
- YES
- ONLY
- REGISTER
- ALL
- NONE
- ddname
- RESETRTS

1 Not valid with BACKOUT or Instant Snapshot copies

Figure 23: RECOVER syntax—TORBA or TOLOGPOINT specification

TORBA or TOLOGPOINT specification

- TORBA
- TOLOGPOINT
- LASTQUIESCE
- LASTCOMMONQ
- LASTARCHQ
- LASTSHUTDOWN
- LOGMARK logMarkName
- logPoint
- BACKOUT
- YES
- (relativeGenerationNumber)
- (logMarkGeneration)
Figure 24: RECOVER syntax—TOCOPY specification

TOCOPY specification

- TOCOPY
- LASTCOPY
  - (relativeGenerationNumber)
  - dataSetName
- TOVOLUME
  - volumeSerialNumber
- CATALOG
  - TOSEQNO
  - integer

* Not valid with OBJECTSET

Figure 25: RECOVER syntax—OUTCOPY specification, continued

OUTCOPY specification, continued

- OUTCOPYDDN specification
- RECOVERYDDN specification
- TAG tagName
- SPACE
  - (pri.sec)
  - CYL
  - TRK
Figure 26: RECOVER syntax—OUTCOPYDDN specification

OUTCOPYDDN specification

```
OUTCOPYDDN — ( DDName1, DDNamePrefix1, outputDescriptor1, DDName2, DDNamePrefix2, outputDescriptor2 )
```

1. These default values apply to table spaces with fewer than 100 partitions. If the table space has 100 partitions or more, the corresponding default values are BMCCY and BMCCZ.
2. These values are the default values used by the OUTPUT command.
3. These values are used to specify the OUTPUT command, which will control the allocation of the copy.

Figure 27: RECOVER syntax—RECOVERYDDN specification

RECOVERYDDN specification

```
RECOVERYDDN — ( DDName3, DDNamePrefix3, outputDescriptor3, DDName4, DDNamePrefix4, outputDescriptor4 )
```

1. These default values apply to table spaces with fewer than 100 partitions. If the table space has 100 partitions or more, the corresponding default values are BMCRY and BMCRZ.
2. These values are the default values used by the OUTPUT command.
3. These values are used to specify the OUTPUT command, which will control the allocation of the copy.
### RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE command dependencies and prohibitions

The following table describes the common dependencies for coding the RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands.

**Table 5: RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE command dependencies**

<table>
<thead>
<tr>
<th>Option coded</th>
<th>Option that must be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCOPY</td>
<td>TOCOPY LASTCOPY</td>
</tr>
<tr>
<td></td>
<td>OR</td>
</tr>
<tr>
<td></td>
<td>TOCOPY data set name where data set name matches the last DSNAME in INCOPY</td>
</tr>
<tr>
<td></td>
<td>OR</td>
</tr>
<tr>
<td></td>
<td>RBA/LOGPOINT on the last INCOPY specification</td>
</tr>
<tr>
<td>OBIDXLAT and INCOPY</td>
<td>TOCOPY on the RECOVER TABLESPACE command statements</td>
</tr>
<tr>
<td></td>
<td>OR</td>
</tr>
<tr>
<td></td>
<td>supply the source DBID and PSID in the OBIDXLAT clause</td>
</tr>
<tr>
<td>FROMRBA or FROMLOGPOINT</td>
<td>LOGAPPLY ONLY</td>
</tr>
<tr>
<td>BACKOUT</td>
<td>TORBA or TOLOGPOINT</td>
</tr>
<tr>
<td>INCOPY and OBJECTSET</td>
<td>MODEL with symbolics</td>
</tr>
<tr>
<td>INDEX(ALL) and INDEP</td>
<td>MODEL with symbolics</td>
</tr>
<tr>
<td>OUTSPACE</td>
<td></td>
</tr>
</tbody>
</table>

Table 6 on page 165 describes the common prohibitions to coding the RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands.
### Table 6: RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE command prohibitions

<table>
<thead>
<tr>
<th>Option coded</th>
<th>Option that must not be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOCOPY</td>
<td>LOGAPPLY ONLY OR LOGONLY</td>
</tr>
<tr>
<td>INCOPY or INLOG</td>
<td>LOGAPPLY ONLY OR LOGONLY</td>
</tr>
<tr>
<td>INCOPY and OBIDXLAT without specifying the source IDs and do NOT code DROPRECOVERY</td>
<td>RBA or LOGPOINT on any INCOPY specification</td>
</tr>
<tr>
<td>DROPRECOVERY</td>
<td>INDEP OUTSPACE</td>
</tr>
<tr>
<td>BACKOUT</td>
<td>OUTCOPY ONLY, OBIDXLAT, or DROPRECOVERY</td>
</tr>
</tbody>
</table>

### Additional RECOVER INDEX and RECOVER INDEXSPACE limitations

Additional limitations apply when you use RECOVER INDEX or RECOVER INDEXSPACE to recover indexes.

Consider the following items in addition to those listed in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE command dependencies and prohibitions” on page 164.

**Note**

An index space does not have to be defined with the COPY YES attribute to be recoverable.

- If you have used the BMC PACLOG for DB2 utility to exclude the index log records, index recovery from logs is not possible.

- If you specify RECOVER INDEX (ALL) or RECOVER INDEXSPACE (ALL), RECOVER INDEX and RECOVER INDEXSPACE do not support the FROMRBA/FROMLOGPOINT options and only support the OUTCOPY option when the copies are dynamically allocated. (For syntax regarding dynamic allocation, see “OUTPUT command” on page 142.) You can use separate index specifications with RECOVER INDEX and RECOVER INDEXSPACE to use those options.

- If you have used an ALTER statement to change the PIECESIZE for an index, index recovery from log is not possible.
- If a table space is recovered to a prior point in time, the default TORBA or TOLOGPOINT value for recovery of any indexes on that table space is the log point specified for the table space. If you specify different log points for the table space and its index, either with TORBA, TOLOGPOINT, or TOCOPY, NGT Recover ends with an error.

- If you recover a table space by using the TOCOPY and the INCOPY keywords, and you also recover a related index by using the TOCOPY and the INCOPY keywords, NGT Recover cannot verify that the two copies are in sync.

- NGT Recover does not prevent a point-in-time recovery of an index that does not include a point-in-time recovery of the table space or other indexes on the table space.

- Indexes must always have a copy or other backup if RECOVER INDEX or RECOVER INDEXSPACE is used without BACKOUT. If an index is created on a table that contains data, DB2 does not log the index updates. Similarly, index rebuilds or updates that result from LOG YES utilities are not logged. If the index is created on a table, an image copy or other backup must be made before the index is recoverable.

- In addition to the LOG NO SYSCOPY events that make a table space unrecoverable, any LOAD LOG YES event or REORG LOG YES event between the image copy and the TORBA or TOLOGPOINT value makes an index unrecoverable. REBUILD INDEX and REORG INDEX command statements can also render an index unrecoverable, and NGT Recover cannot detect these events for COPY NO indexes if non-BMC utilities are used.

NGT Recover REBUILD INDEX events are registered in the BMCXCOPY table for all COPY NO indexes. BMC REORG PLUS registers REORG INDEX events for COPY NO indexes in the BMCXCOPY table.

You cannot use RECOVER INDEX or RECOVER INDEXSPACE commands for forward index recovery through a point-in-time recovery event created by a BACKOUT job.

- If you specify the INCOPY option with OBIDXLAT, and you do not specify INDEP OUTSPACE or OUTCOPY, a warning message stating that a copy is required is issued and the original status of the index and its table space is not restored.

---

**Note**

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

---

- When you use RECOVER INDEXSPACE with OBIDXLAT for indexes, you must also specify the table ID that is associated with the index as an additional
OBID(x,y) clause, where x is the ID of the source table and y is the ID of the target table. If you do not specify this extra OBID clause, the job issues the following message:

```
BMC407641 TABLE OBID 'xxxx' FOUND AT OFFSET '3A' IN INDEX HEADER WILL NOT BE TRANSLATED
```

You may, however, omit both OBID(x,y) clauses. If you omit both clauses, appropriate default values are supplied. For more information, see “OBIDXLAT specification” on page 191.

## RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE option descriptions

This topic describes the options available with RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE. Not all of the options described are valid in a single RECOVER TABLESPACE, RECOVER INDEX, or RECOVER INDEXSPACE command. The recovery strategy that best meets your needs determines which options you should include in a single command statement.

### RECOVER

Use the RECOVER command when you want to perform an actual recovery.

### SIMRCVR

SIMRCVR is a synonym for the RECOVER command.

Use the SIMRCVR command to clarify that you are running in simulation mode. When you use SIMRCVR, you must also specify OPTIONS SIMULATE YES, which requires a valid BMC Recovery Management for DB2 solution password. For more information, see “SIMULATE YES” on page 132.

The SIMRCVR command generally has the same meaning and the same syntax in simulation mode as a regular RECOVER command in non-simulation mode. However, the following limitations apply to SIMRCVR:

- You cannot use any of the following options:
  - TOCOPY
  - TOLOGPOINT
  - TORBA
Even though you cannot specify TOLOGPOINT or TORBA on the SIMRCVR command statement, if you are using the BMC Recovery Management for DB2 solution and have a valid password for the solution, you can simulate a recovery to a previous point in time by coding SIMULATE YES LOGPOINT X'logPoint' on the OPTIONS command statement.

---

**Table space specification**

For option descriptions for specifications which are included in the table space specification, see:

- “OBIDXLAT specification” on page 191
- “Non-registered copy or INLOG specification” on page 188
- “FROMRBA or FROMLOGPOINT specification” on page 180
- “OUTCOPY specification” on page 193

The syntax diagram for the table space specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**TABLESPACE databaseName.tablespaceName**

The TABLESPACE option specifies the table space to be recovered.

- **databaseName** is the name of the database containing the table space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database that is defined with TYPE='W' in SYSIBM.SYSDATABASE.

- **tableSpaceName** is the name of the table space within the named database. BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

**OBJECTSET specification**

Use the OBJECTSET specification with TABLESPACE to recover only the table spaces that are in a defined group.

For more information, see “OBJECTSET specification” on page 177 and “Using BMC RECOVERY MANAGER groups” on page 518.

---

**Note**

To include the indexes in a recovery using TABLESPACE OBJECTSET, specify INDEXES YES.
INDEXES

The INDEXES option allows you to specify that you want NGT Recover to recover the indexes associated with the table space(s) given by the TABLESPACE option of the RECOVER command. The default is INDEXES NO indicating no indexes recovery.

**Note**
The use of INDEX is synonymous to INDEXES for this option.
The INDEXES option is not applicable to INDEXSPACE or INDEX specifications.

INDEXES NO

Specifying INDEXES NO tells NGT Recover to not recover the indexes for the specified table space or table spaces.

INDEXES YES

Specifying INDEXES YES tells NGT Recover to recover all indexes associated with the table space(s) specified by the TABLESPACE.

**Note**
INDEXES YES is invalid with an unqualified OBJECTSET specification (OBJECTSET without TABLESPACE).

DSNUM

DSNUM specifies a single data set in the specified table space or the entire table space for recovery.

**DSNUM ALL**

DSNUM ALL is the default and specifies that all of the data sets in the table space are to be recovered. If the table space is nonpartitioned and DSNUM ALL is specified or implied, image copies made for specific data sets will only be considered for recovery if they are registered at the same log point.

**DSNUM integer**

Use this option to recover only one partition or data set in the table space. For partitioned table spaces, DSNUM integer specifies the number (from 1 through 4096) of a single partition in the specified table space. For nonpartitioned table spaces, DSNUM integer specifies the number (from 1 to 32) of a single data set of the specified table space.
DSNUM `begin : end`

Use this option to recover a range of partitions in the table space. For partitioned table spaces, `begin` specifies the number (from 1 through 4095) of the first partition in the range and `end` specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

AUX

The AUX option specifies if auxiliary objects will be included with the recovery of the base table spaces. For more information, see “AUX” on page 117.

RICHK

RICHK specifies if NGT Recover will check for referential integrity constraints on the objects being recovered.

RICHK YES

RICHK YES checks for referential integrity constraints and sets check pending (CHKP) status for dependent objects not recovered in the same step to the same PIT.

RICHK NO

RICHK NO does not check for referential integrity. NO is the default value.

DROPRECOVERY

DROPRECOVERY specifies that you want to recover a dropped table space. The DROPRECOVERY keyword requires either INCOPY, INLOG, LOGONLY, or LOGAPPLY ONLY. When DROPRECOVERY is coded, the utility attempts to locate the point at which the table space was re-created and uses that point as a stopping point or verifies it against the TORBA or TOLOGPOINT value. If the space was dropped and re-created several times, you must specify a TORBA or TOLOGPOINT value in the NGT Recover job to tell NGT Recover when to stop applying log records. Use DSN1LOGP to identify the log point of the first CREATE TABLESPACE statement after the drop. This is the log point that you should specify with the TORBA or TOLOGPOINT option. If you do not specify a TORBA or TOLOGPOINT value in this case, the recovery will result in an empty table space.

If the dropped space was versioned, you must recreate the space with the same versions, or in some cases, you can use DB2 REPAIR VERSIONS.
■ If the dropped space has been reorganized after the first alter, you can recreate the space to match the latest version of the space, recover the space, and then run DB2 REPAIR VERSIONS. You cannot rebuild indexes until after you run DB2 REPAIR VERSIONS.

■ If the dropped space has not been reorganized, you must recreate the space as it existed originally, before any alters, perform the alters to create the same versions that the dropped space had, and then recover the space.

**WARNING**
Use DROPRECOVERY only if the *table space* has been dropped (not just a *table* within the *table space*).

**Note**
DROPRECOVERY is not valid for index recovery.

For examples on how to use DROPRECOVERY, see “Recovering a dropped table space or table” on page 525.

**TRANSFORM**

Use TRANSFORM to specify that you want to do a DDL transformation. A TOCOPY LASTCOPY and OBID translation is assumed.

**Note**
TRANSFORM is part of the High-speed Structure Change process. SHRLEVEL CHANGE TRANSFORM requires a Recovery Management solution password, but a SHRLEVEL REFERENCE TRANSFORM does not.

The High-speed Structure Change process allows you to perform the following transformations:

■ Simple table space to partition-by-growth (PBG) table space
■ Non-large partitioned table space to large partitioned table space
■ Range-partitioned table space to range-partitioned (PBR) universal table space
■ Range-partitioned table space to partition-by-growth universal table space
■ Multi-table table space to partition-by-growth universal table space

The High-speed Structure Change process also allows you to transform your indexes, avoiding costly sort CPU that would be required to rebuild the index.

At a more granular level, you can use the High-speed Structure Change process to:

■ Change the segment size (SEGSIZE) of a table space
■ Transform a non-segmented table space into a segmented one
Change the data set size (DSSIZE or PIECESIZE) of your table space or index
- Change the compression attribute of your index
- Add or remove the MEMBER CLUSTER attribute
- Increase or decrease MAXPARTITIONS

Multi-table table space transformations have the following requirements:
- Multi-table table spaces must be segmented.
- Each table must be processed with a separate TRANSFORM statement.
- The TRANSFORM statement must include OBIDXLAT OBID with a source OBID value. The target OBID can be defaulted. The DBID and PSID are not required.
- The target indexes must be rebuilt.

**Note**
The transformation requires the RBA/LRSN format of the source and target objects to match.

**Figure 29 on page 172** provides an example of the syntax used to transform a multi-table table space into two separate single-table table spaces.

**Figure 29: Example syntax for a multi-table table space transformation**

```
RECOVER TABLESPACE PBGDB.TS1 DSNUM ALL
  OBIDXLAT OBID(3,3)
  TRANSFORM
  INCOPY FULL SNAPSHOT
    MODEL RDADB.DSNDBD.MULTDB.MULTTS.SNAP&DSNUM
  REBUILD INDEX(ALL) TABLESPACE PBGDB.TS1 DSNUM ALL

RECOVER TABLESPACE PBGDB.TS2 DSNUM ALL
  OBIDXLAT OBID(7,3)
  TRANSFORM
  INCOPY FULL SNAPSHOT
    MODEL RDADB.DSNDBD.MULTDB.MULTTS.SNAP&DSNUM
  REBUILD INDEX(ALL) TABLESPACE PBGDB.TS2 DSNUM ALL
```

For more information, see the *Recovery Management for DB2 User Guide*.

**Single index specification**

For option descriptions for the OBIDXLAT, Non-Registered Copy, FROMRBA, and OUTCOPY specifications, which are included in the Single index specification, see:
- “OBIDXLAT specification” on page 191
- “Non-registered copy or INLOG specification” on page 188
- “FROMRBA or FROMLOGPOINT specification” on page 180
- “OUTCOPY specification” on page 193
The syntax diagram for the Single index specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**Note**

If INDEXLOG=NO is specified as an installation option or on the OPTION command, single index specifications are not valid on the same RECOVER statement with TABLESPACE. Without TABLESPACE, the parser understands that REBUILD is implied.

**INDEX creatorID.indexName**

The INDEX option specifies the index to be recovered.

- **creatorID** is the qualifier creator ID for the index. The default is the user identifier for the utility.

- **indexName** is the name of the index. Indexes for BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

**OBJECTSET specification**

Use the OBJECTSET specification with INDEX to recover only the indexes that are in a defined group. For more information, see “OBJECTSET specification” on page 177 and “Using BMC RECOVERY MANAGER groups” on page 518.

**INDEXSPACE databaseName.indexSpaceName**

The INDEXSPACE option specifies the index space to be recovered.

- **databaseName** is the name of the database containing the index space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE.

- **indexSpaceName** is the name of the index space within the named database. Indexes for BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

**DSNUM**

DSNUM specifies either a single data set for the specified index or the entire index for recovery.
Note
If you code INDEXLOG=NO in the installation options or on the OPTION command, RECOVER INDEX is analogous to REBUILD INDEX and you should use PART (“PART” on page 226) rather than DSNUM. Otherwise, NGT Recover ends, issues BMC40442 and BMC40443, and requests validation of the options.

DSNUM ALL

DSNUM ALL is the default and specifies that all of the data sets for the index are to be recovered.

If the index is nonpartitioned and DSNUM ALL is specified or implied, only image copies made for the entire index (those registered with DSNUM 0) are used for recovery, with the following exception: If DSNUM ALL is specified and NGT Recover detects that one or more DSNUM n copies are registered for an index at the same RBA, NGT Recover expands the request from DSNUM ALL to DSNUM n, (where n is the number of image copies registered together) and issues an informational message.

For multi-data-set, nonpartitioned index spaces, if the index copy used for recovery is a DSN1COPY, DSNUM ALL is not allowed.

DSNUM integer

Use this option to recover only one partition or data set for an index. For partitioned indexes, DSNUM integer specifies the number (from 1 through 4096) of a single partition of the specified index. For nonpartitioned indexes, DSNUM integer specifies the number (from 1 to 4096) of a single data set for the specified index.

DSNUM begin : end

Use this option to recover a range of partitions in the table space. For partitioned table spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

NOWORKDDN

The NOWORKDDN option causes the keys to be sent directly to the sort routine without first writing these keys to a work data set. With RECOVER INDEX, NOWORKDDN is only valid when you have also specified INDEXLOG AUTO.
Multiple index specification

For option descriptions for the OUTCOPY specification, which is included in the Multiple index specification, see:

■ "OUTCOPY specification" on page 193

The syntax diagram for the Multiple index specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

Note

If INDEXLOG=NO is specified as an installation option or on the OPTIONS command, multiple index specifications are not valid on the same RECOVER statement with TABLESPACE. Without TABLESPACE, the parser understands that REBUILD is implied.

INDEX creatorID.indexName

The INDEX option specifies the index to be recovered as follows:

■ creatorID is the qualifier creator ID for the index. The default is the user identifier for the utility.

■ indexName is the name of the index. Indexes for BMCUTIL, BMCSYNC, and BMXCCOPY (tables that are used by BMC utilities) cannot be used.

OBJECTSET specification

Use the OBJECTSET specification with INDEX to recover only the indexes that are in a defined group. For more information, see “OBJECTSET specification” on page 177 and “Using BMC RECOVERY MANAGER groups” on page 518.

INDEX (ALL) TABLESPACE databaseName.tableSpaceName

(ALL) TABLESPACE specifies that all indexes for the named table space are to be recovered. The parentheses around ALL are optional.

TABLESPACE specifies the table space from which all indexes are to be recovered as follows:

■ databaseName is the name of the database containing the table space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE.
- `tableSpaceName` is the name of the table space within the named database. BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

**AUX**

The AUX option specifies if auxiliary objects will be included with the recovery of the base table spaces. For more information, see “AUX” on page 117.

**INDEXSPACE** `databaseName.indexSpaceName`

The INDEXSPACE option specifies indexes to be recovered as follows:

- `databaseName` is the name of the database containing the index space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE.

- `indexSpaceName` is the name of the index space within the named database. Indexes for BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

**INDEXSPACE (ALL) TABLESPACE** `databaseName.tableSpaceName`

(ALL) TABLESPACE specifies that all indexes for the named table space are to be recovered. The parentheses around ALL are optional.

TABLESPACE specifies the table space from which all indexes are to be recovered as follows:

- `databaseName` is the name of the database to which the table space belongs. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE. The default is DSNDB04.

- `tableSpaceName` is the name of the table space whose indexes are to be recovered. BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

If you do not specify the TABLESPACE option, NGT Recover determines the table space for the first valid index specified. All other specified indexes must belong to the same table space.

**NOWORKDDN**

The NOWORKDDN option causes the keys to be sent directly to the sort routine without first writing these keys to a work data set. With RECOVER INDEX, NOWORKDDN is only valid when you have also specified INDEXLOG AUTO.
**OBJECTSET specification**

The OBJECTSET option is available in RECOVER statement syntax to recover groups that are defined using RECOVERY MANAGER.

The OBJECTSET specification is shown in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**OBJECTSET creator.name**

OBJECTSET is available in the following syntactical forms:

- **RECOVER OBJECTSET creator.name**
  RECOVER OBJECTSET recovers all objects in a group, both table spaces and indexes.

  __Note__
  INDEXES YES is not valid with a RECOVER OBJECTSET specification.

- **RECOVER TABLESPACE OBJECTSET creator.name**
  RECOVER TABLESPACE OBJECTSET recovers only the table spaces in the group.

- **RECOVER TABLESPACE OBJECTSET creator.name INDEXES YES**
  RECOVER TABLESPACE OBJECTSET... INDEXES YES recovers the table spaces in the group along with their associated indexes.

- **RECOVER INDEX OBJECTSET creator.name**
  RECOVER INDEX OBJECTSET recovers only the indexes in the group.

  __Note__
  SIMRCVR is also valid in place of RECOVER.

The following rules apply to creator.name:

- **creator** specifies the name of the creator of the group and can have up to 128 characters in length.

- **name** specifies the group name and can have up to 128 characters in length.

- You can delimit both creator and name with single or double quotation marks.

- Both creator and name can contain the special characters $, #, and @ in any position.
NGT Recover uses dynamic grouping to determine the group contents at the time that you run the NGT Recover job containing the OBJECTSET option. Because NGT Recover reads the objects in the group each time the job is executed, objects may be added or removed from the group.

**Note**

NGT Recover does not read RECOVERY MANAGER recovery options for the group. If RECOVERY MANAGER recovery options change, you must run RECOVERY MANAGER to pick up the new options and generate the control cards.

Following are restrictions that apply when you use OBJECTSET in any of its forms on the RECOVER statement:

- AUX is ignored with any specification of OBJECTSET.

- You cannot specify DSNUM with OBJECTSET. NGT Recover uses the group definition for the object.

- You can specify OBIDXLAT with OBJECTSET only when all the OBID specifications can be omitted. For information about OBIDXLAT, see **OBIDXLAT specification on page 191**.

- Only TOCOPY LASTCOPY is valid with OBJECTSET. (You cannot use a specific data set name.)

- You cannot use INLOG with OBJECTSET.

- NGT Recover allows only one OBJECTSET clause, and the statement can not contain additional TABLESPACE, INDEX, or INDEXSPACE clauses.

When you use OBJECTSET, if NGT Recover tries to recover a group of objects and more than 10 of the objects are unrecoverable, NGT Recover issues an error for each object that is unrecoverable and discontinues further processing of the group. This behavior is based on the ERRCONT option that has a default value of 10. To avoid this behavior, set the ON ERROR CONTINUE option with a value that will exceed the possible number of errors in the group, and the recovery will complete.

**EXCLUDE (dbname.ts or qualifier.ix)**

Use the EXCLUDE option after an OBJECTSET specification to exclude one or more objects from recovery. You can use wildcards or fully qualified names to specify the exclusions. The following wildcards are valid:

- % and * match no characters to any number of characters in the object name.

- _ and ? match a single character in the object name.
List the excluded objects following the EXCLUDE keyword. Each item in the list must be in the form dbname.ts or qualifier.ix and you must separate the individual items with commas. Enclose the list in parentheses.

The following considerations apply to use of the EXCLUDE option with the OBJECTSET specification:

- When you specify INDEXES YES for a RECOVER OBJECTSET statement and specify a tablespace for exclusion, the associated indexes are also excluded.
- You cannot exclude individual indexes when INDEXES YES is specified.

**DSNUM**

DSNUM specifies a single data set in the specified table space or the entire table space for recovery.

**DSNUM ALL**

DSNUM ALL is the default and specifies that all of the data sets in the table space are to be recovered.

**DSNUM integer**

Use this option to exclude only one partition or data set in the table space. For partitioned table spaces, DSNUM integer specifies the number (from 1 through 4096) of a single partition in the specified table space. For nonpartitioned table spaces, DSNUM integer specifies the number (from 1 to 32) of a single data set of the specified table space.

**DSNUM begin : end**

Use this option to exclude a range of partitions in the table space. For partitioned table spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

**OBIDXLAT**

Use the OBIDXLAT keyword to change the internal IDs in the DB2 table space or index as the table space or index is recovered.
Note

Note the following information about the use of OBIDXLAT:

- When you use Instant Snapshots for migration and specify the OBIDXLAT option, NGT Recover automatically detects if the OBIDs have not changed and avoids the overhead of further processing.

- OBIDXLAT of Instant Snapshots of universal table spaces (UTS) further reduces the processing time, even if OBIDs change.

- If you specify RESET, NGT Recover does not get a speed improvement. For data sharing environments, RESET is usually not necessary. For non-data-sharing environments, if the RBAs on the target system are larger than the RBAs on the source, RESET is probably not necessary.

RESET

The RESET keyword causes the log points in each data page to be reset to 0. You must specify RESET when you are migrating data from one DB2 non-data-sharing system to another because the log point values on the first subsystem is meaningless or misleading on the target DB2.

The recovery log point and level fields in the header page are also reset to 0 if you are not using the INDEPENDENT OUTSPACE option. If you are using the INDEPENDENT OUTSPACE option, the level fields are reset but the recovery log point is not.

FROMRBA or FROMLOGPOINT specification

You can specify FROMRBA or FROMLOGPOINT only with LOGAPPLY ONLY. When you use LOGAPPLY ONLY, you must specify FROMRBA or FROMLOGPOINT for each table space and index specification.

FROMRBA or FROMLOGPOINT specifies the log point in the log where application of the log records should begin for the object or object data set being recovered. You can specify the log point with a keyword or with a hexadecimal string.

This option assumes that the object or objects have been brought to an appropriate point by some other activity prior to this step.

You can use the FROMRBA and FROMLOGPOINT keywords interchangeably, regardless of the version of DB2 that you are using.

The syntax diagram for the FROMRBA or FROMLOGPOINT specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.
FROMRBA LASTQUIESCE or FROMLOGPOINT LASTQUIESCE

FROMRBA LASTQUIESCE or FROMLOGPOINT LASTQUIESCE specifies that the log point where application of the log records will begin for this object (or object data set) is the log point of the most recent quiesce point registered in SYSIBM.SYSCOPY. If this is recovery of an index, the QUIESCE entry for the table space is used. You can abbreviate LASTQUIESCE to LASTQ.

Note
If you use LASTQUIESCE and the quiesce is prior to the ALTER ADD PART, the recovery will fail. You must specify a hard-coded RBA. If the partition was added for a partition-by-growth universal table space, you can use the LASTQUIESCE option.

FROMRBA LASTCOPY or FROMLOGPOINT LASTCOPY

FROMRBA LASTCOPY or FROMLOGPOINT LASTCOPY specifies that the log point where application of the log records will begin for this object (or object data set) is the log point of the most recent image copy (full or incremental) registered.

FROMRBA LASTARCHQ or FROMLOGPOINT LASTARCHQ

FROMRBA LASTARCHQ or FROMLOGPOINT LASTARCHQ specifies that the log point of the most recent ARCHIVE LOG MODE(QUIESCE) command is the point where application of the log records will begin for this object (or object data set).

FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN

FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN specifies that the log point of the most recent successful STOP DB2 command is the point where application of the log records will begin for this object (or object data set).

Note
If you specify this keyword in a data sharing environment, log records since the last successful shutdown of the member subsystem on which the job is running are used for recovery, without regard to other member subsystems. For this reason, use extreme caution when you use FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN in a data sharing environment to ensure that the shutdown used is really a point consistent with the state of the spaces.

FROMRBA X'logPoint' or FROMLOGPOINT X'logPoint'

FROMRBA X'logPoint' or FROMLOGPOINT X'logPoint' specifies the log point in the log where application of the log records for this object will begin.
'logPoint' is a string of up to twelve hexadecimal digits.

**INCOPY specification**

The syntax diagram for the INCOPY specification is in

- “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151

**DSNAME dataSetName**

DSNAME *dataSetName* is the data set name of the copy used during recover.

For an Instant Snapshot copy or an IBM FlashCopy, this name should be the VSAM data component name or the cluster component name. The VSAM data component name is printed in the output from the BMC Next Generation Technology Copy for DB2 for z/OS job that created the Instant Snapshot. For a FlashCopy, the cluster component name is printed in the output from the DB2 COPY utility. In reporting, NGT Recover displays the name that was supplied with this parameter.

If you specify a generation data group (GDG) name, you can either specify the relative GDG name or the fully qualified data set name.

If you specify a relative GDG name for an existing data set, use either (0) or (-n) for the relative generation. Do not refer to it as the (+1) generation. The most recent data set is the (0) generation, even if it was created in a previous step in the same job.

**MODEL dataSetName**

MODEL *dataSetName* allows you to construct the copy data set by using symbolic variables. This is useful when used with OBJECTSET processing.

*Note*

When you use INCOPY with TRANSFORM, specify the VSAM data component instead of the VSAM cluster. *dataSetName* must include a node for the data set number represented by A&LDSNUM. This node can appear anywhere within *dataSetName*.

You can construct *dataSetName* by using the symbolic variables in “Using symbolic variables” on page 494.

**DSSIZE (integerG)**

DSSIZE is used with TRANSFORM to specify the data set size of the source table space. In most cases, the default value should be sufficient.
**Note**

DSSIZE and TRANSFORM are part of the High-speed Structure Change process.

Valid values are 0, 1G, 2G, 4G, 8G, 16G, 32G, 64G, 128G, and 256G. If the table space is not partitioned, the default is 2G. If the table space is partitioned and not a LARGE table space (has a 4-byte RID), the DSSIZE is calculated based on the number of partitions. If the table space is a LARGE type, the data set size is read from the header page. If the DSSIZE is not stored in the header page, you must specify DSSIZE in the SYSIN.

For more information, see the *Recovery Management for DB2 User Guide*.

**PIECESIZE (...)**

Use PIECESIZE with TRANSFORM to specify the data set size of the source nonpartitioning index space. In most cases, the default value should be sufficient.

**Note**

PIECESIZE and TRANSFORM are part of the High-speed Structure Change process.

Valid values are any integer that is a power of 2 between 256K through 64G. The PIECESIZE is calculated based on the value of ptshift, which is found in the header page. If ptshift is not valued (an index migrated from an early DB2 release), you must specify PIECESIZE.

For more information, see the *Recovery Management for DB2 User Guide*.

**RBA X'logPoint' or LOGPOINT X'logPoint'**

RBA X'logPoint' or LOGPOINT X'logPoint' indicates the log point of the copy. The keywords RBA and LOGPOINT are interchangeable, regardless of the version of DB2 used.

To recover to the current point or to a quiesce point, you must code a log point value on the last copy specified so that NGT Recover knows at what point in the log to start obtaining log records for the table space or index.

If no log point is specified on the last copy, you must recover TOCOPY and no log is processed.

**SHRLEVEL REFERENCE**

SHRLEVEL REFERENCE indicates that the share level of the copy is REFERENCE. If you do not specify a value for SHRLEVEL, the default is REFERENCE.
SHRLEVEL CHANGE

SHRLEVEL CHANGE indicates that the share level of the copy is CHANGE.

ENCRYPTED

Use the ENCRYPTED option to specify that the specified copy is an encrypted image copy made by NGT Copy. For more information about using encrypted copies, see “Recovering encrypted copies” on page 506.

Note

Encryption is a feature of the BMC Recovery Management for DB2 solution and requires a valid Recovery Management solution password.

TIMESTAMP timestamp

The TIMESTAMP option specifies the timestamp when the encrypted copy was registered in BMCXCOPY. The timestamp value determines which key from the key data set (specified by the KEYDSNAM installation option described on “KEYDSNAM= keyDataSetName” on page 586) NGT Recover uses to decode the copy. The timestamp value uses either of the following formats:

yyyy-mm-dd-hh.mm.ss

yyyy-mm-dd-hh.mm.ss.nnnnnn

You can omit leading zeros from the month, day, or hour parts of the timestamp; you can omit trailing zeros from the microseconds part of the timestamp.

COMPRESSED

Use the COMPRESSED option to indicate that the specified image copy is a compressed copy of a compressed index made by DSN1COPY or NGT Copy. For more information about NGT Copy compressed copies of compressed indexes, see the BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual section "Copying compressed indexes."

INVOLUME

The INVOLUME specification gives the volume information for the copy indicated in the INCOPY specification.
INVOLUME CATALOG

INVOLUME CATALOG is the default and identifies the image copy data set as a cataloged data set. It specifies that the volume serial number should be obtained from the operating system catalog.

INVOLUME volSerialNumberList

INVOLUME volSerialNumberList identifies the image copy data set as uncataloged where volSerialNumberList provides the serial numbers of the volumes containing the image copy data set. Include all volume serials in the order needed. volSerialNumberList specifies a list of one or more volume serial numbers, separated by commas.

INDEVT deviceType

The deviceType parameter for INDEVT identifies the device on which the copy resides. It is required for uncataloged data sets.

INSEQNO integer

The integer parameter for INSEQNO is the sequence number of the data set containing the image copy data set on the tape volumes. If you specify INCOPIY and INVOLUME volumeSerialNumber (to indicate an uncataloged data set) but do not specify INSEQNO, NGT Recover assumes file sequence number one.

INDEPENDENT OUTSPACE specification

Use the INDEPENDENT (or INDEP) OUTSPACE option to redirect the output of the recovery to a data set that is not used by DB2.

The syntax diagram for the INDEPENDENT OUTSPACE specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

INDEPENDENT OUTSPACE or INDEP OUTSPACE

INDEP OUTSPACE recoveries do not require you to stop the spaces that are being recovered, and they do not interfere with production processing. INDEP OUTSPACE enables you to do the following things:

- Test a recovery process without taking spaces offline
- Migrate data to another DB2 system
- Recover an image to another data set while allowing access to data that is experiencing intermittent errors
For DB2 objects that have multiple underlying VSAM data sets, the following rules apply:

- If the object being recovered is specified by using DSNUM ALL (either explicitly or by default), the output of the recovery is redirected to multiple independent data sets.

- If only a single data set or partition of an object is being recovered (by using DSNUM integer), the output of the recovery is redirected to a single independent data set.

You can specify the name of each independent data set or you can accept the NGT Recover model as a default.

**Note**

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

You can preallocate the INDEP OUTSPACE data sets using the following guidelines:

- For VCAT-defined spaces, ensure that the volumes or space allocated is large enough to hold the data set.

- If the source space is defined using a STOGROUP, and if primary and secondary quantities are not defined for the source space, NGT Recover uses a sliding-scale calculation for secondary extents, similar to the method DB2 uses.

For more information, see “Primary space allocation” on page 81 and “Secondary extents” on page 81.

You can allow NGT Recover to dynamically allocate the INDEP OUTSPACE data sets for STOGROUP-defined spaces. If the INDEP OUTSPACE data set does not exist, NGT Recover uses the attributes of the source space to dynamically allocate the target INDEP OUTSPACE data set. If the INDEP OUTSPACE data set exists, NGT Recover uses it as is (no DELETE or DEFINE executed).

NGT Recover includes support for multi-data-set spaces. For example, NGT Recover will dynamically allocate A002, A003, to A00n INDEP OUTSPACE data sets as needed.

To have NGT Recover dynamically allocate the INDEP OUTSPACE data set on a specific volume or volumes, pass a volume list by using the STOGROUP...USEORDER specification in the OPTIONS statement (see
NGT Recover statements that use STOGROUP... USEORDER are shown in the following example:

```sql
OPTIONS STOGROUP TESTSTO USEORDER(NEW004,NEW005,NEW006)
RECOVER TABLESPACE TESTDB.TESTTS
    INDEP OUTSPACE
```

The default model name is identical to the DB2 name of the object being recovered, but with BMCDBC substituted for DSNDBC in the CLUSTER portion of the VSAM data set:

```sql
vcat.BMCDBC.databaseName.objectName.p0001.znnn
```

The variables are defined as follows:

- **vcat** is the ICF catalog name.
- **databaseName.objectName** is the name of the database and object containing the data set being recovered.
- **p** corresponds to the value that is used by DB2 for this node.
- **znnn** is the data set number.

The letters correspond to the first digit—0, 1, 2, 3, or 4—of the partition number. `nnn` represents the rest of the partition number. `znnn` represents the partitions numbers as follows:

- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E096 for partitions 4000 through 4096

When you want to specify a name of your own choosing, use the MODEL option to specify the name of the CLUSTER portion of the VSAM data set that you want to use. The data set name that you specify may include any of the symbolic variables that are listed in the description of the MODEL keyword in the following section.

When you use INDEP OUTSPACE, you must specify it after the object specification, as shown in the syntax diagram in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151. Also, see “Testing recovery with simulation” on page 52 and “Examples of NGT Recover jobs” on page 367 for information about applications of this feature.

**MODEL**

The MODEL option specifies the non-DB2 data set cluster name to which you want to redirect the output from the recovery of the specified object. The data
set name that you specify may optionally include the symbolic variables “Using symbolic variables” on page 494.

**LOGSORT specification**

For a description of the LOGSORT specification options, see

- “LOGSORT work data set specification description” on page 108

The syntax diagram for the LOGSORT specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**Non-registered copy or INLOG specification**

The INCOPY clause recovers by using non-registered image copies.

The syntax diagram for the Non-Registered Copy or INLOG Use specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**INCOPY FULL**

Specify the INCOPY clause to recover by using non-registered image copies. You must specify a full image copy. You may specify any number of incremental copies.

*Note*

When you specify INCOPY, NGT Recover analyzes relevant events in SYSCOPY and BMCXCOPY, unless you specify TOCOPY LASTCOPY (see “TOCOPY specification” on page 202). Then, no access to SYSCOPY or BMCXCOPY occurs.

INCOPY is recommended as the technique to specify a copy of an index made with DSN1COPY for use in an index recovery. Specification of the LOGPOINT and SHRLEVEL values are optional. If the index is encrypted, the ENCRYPTED option (”INCOPY specification” on page 182) is required.

If this recovery is not an INDEPENDENT OUTSPACE recovery or an OUTCOPY ONLY recovery, NGT Recover will register a point-in-time recovery with a PIT_RBA of zeros. Further, unless registered output copies are being created, COPY PENDING is set on any table space so recovered.
Note the following information about the use of INCOPY:

- INCOPY does not support the use of a cabinet copy created by the BMC Next Generation Technology Copy for DB2 for z/OS product. If you have only a cabinet copy and need to do a drop recovery, you can use the NGT Copy COPY IMAGECOPY command to unstack a cabinet copy and register the copies in SYSCOPY. You can then use the individual copies with the INCOPY option.

- NGT Recover supports Online Consistent Copy and consistent FlashCopy with log apply when you do not specify INCOPY.

- NGT Recover supports Online Consistent Copy and consistent FlashCopy with INCOPY TOCOPY specified.

- NGT Recover does not support Online Consistent Copy or consistent FlashCopy when you specify both INCOPY and log apply (RBA or LOGPOINT specified in the INCOPY specification).

**INLINE**

INLINE specifies that the copy specified in the INCOPY specification is an inline copy created by COPYDDN or RECOVERYDDN from a DB2 REORG or LOAD utility. To use an inline copy, you must specify this parameter. This option is valid for full image copies only. The INLINE option is not valid with indexes.

**SNAPSHOT**

SNAPSHOT specifies that the copy specified in the INCOPY specification is an Instant Snapshot copy that is not registered in BMCXCOPY. This option is valid for full image copies only. (For more information, see “Using Instant Snapshots” on page 469.)

**FLASHCOPY**

FLASHCOPY specifies that the copy specified in the INCOPY specification is an IBM FlashCopy. This option is valid for full image copies only.

**INCR**

INCR specifies that the copy is an incremental copy. The INCR copies are processed in the order in which you specified them. Specification of the RBA/LOGPOINT and SHRLEVEL values (see “INCOPY specification” on page 182) are optional.
NOCOPYPEND

If you specify the NOCOPYPEND option, NGT Recover resets COPY-pending status and issues message BMC96232 to inform you that COPY-pending status has been reset even though the space is not recoverable.

For a point-in-time recovery, if you do not specify the NOCOPYPEND option, NGT Recover sets COPY-pending status when all of the following conditions exist:

- You did not specify INDEP OUTSPACE.
- You did not specify OUTCOPY ONLY.
- There is no registered OUTCOPY.

You may want to use the NOCOPYPEND option if you are migrating data from one DB2 subsystem to another in a query or testing system where you do not need to be able to run a recovery on the target space. To migrate the data, you make an image copy on the source system, transport the copy to the target system if necessary, and use RECOVER INCOPY on the target system.

Because the input copy is not registered on the target system, the target space is not recoverable after the RECOVER INCOPY. You cannot run a RECOVER to current on the target system because no usable copies are registered in SYSCOPY or BMCXCOPY.

INLOG RBA X'logPoint' or INLOG LOGPOINT X'logPoint'

INLOG RBA or INLOG LOGPOINT is used when you want to perform recovery by using only log after a LOAD LOG YES or a REORG LOG YES where the event is no longer registered in SYSCOPY. To do so, you must be aware of a point in the log corresponding to the specific event. You must specify a log point that exactly corresponds to such a point; otherwise, the recover will not be able to apply the log and a severe internal error will result with completion code 12.

INLOG RBA or INLOG LOGPOINT specifies the log point in the log where application of the log records should begin for the object being recovered.

'logPoint' is a string of up to twelve hexadecimal digits.

When you use INLOG RBA or INLOG LOGPOINT, you cannot use the TOCOPY option. You also may not specify LOGONLY or LOGAPPLY ONLY. Also, you cannot specify INCOPY when you use INLOG RBA or INLOG LOGPOINT, and any output copies made with the OUTCOPY ONLY facility cannot be registered unless DROPRECOVERY is also specified.

You can use the INLOG RBA and INLOG LOGPOINT keywords interchangeably, regardless of the version of DB2 used.
Note
INLOG RBA and INLOG LOGPOINT are not valid for index recovery because index records are not logged completely for a LOG YES utility.

OBIDXLAT specification

The following rules apply to the OBIDXLAT specification:

- The source or the target DBIDs and PSIDs may be omitted, or the DBID or PSID keyword may be omitted. If a source or target value is omitted from a DBID or PSID clause, the missing value defaults to the DBID or PSID of the object as reflected in the catalog of the current subsystem except when INCOPY is specified without DROPRECOVERY. If INCOPY is specified without DROPRECOVERY, the source DBID and PSID is taken from the image copy.

- For a single-table table space, the source or the target OBID may be omitted, or the OBID keyword may be omitted. If a source or target value is omitted from an OBID clause, the missing value defaults to the OBID of the object as reflected in the catalog of the current subsystem except when INCOPY is specified without DROPRECOVERY. If INCOPY is specified without DROPRECOVERY, the source OBID is taken from the image copy. If more than one OBID clause is coded, the source and target values are required.

- For multiple-table table spaces, rows are not translated for which there is no OBID clause, but you must code at least one OBID clause with both source and target values, or you must code a DBID or PSID clause.

- For an index, if you code an OBID clause, you must code two OBID clauses, one for the index and one for the table on which the index is built. When INCOPY is specified for an index, you may omit both OBID clauses, in which case the source OBID for the index and the table is taken from the image copy. If you do not omit the ID clauses, you must code both the target and source values for both OBID clauses.

If you code INCOPY and OBIDXLAT, you must code TOCOPY or you must supply the source DBID and PSID in the clause. You can use the USEHDROBIDS option (“USEHDROBIDS” on page 119 and “USEHDROBIDS=YES” on page 596) to indicate if the OBIDs in the header are valid or not.

If you code OBIDXLAT and log apply is required, you should also specify OUTCOPY ONLY. These specifications produce an output copy that you can use on a different system with different OBIDs. If you were to do OBIDXLAT to an existing space, the object IDs would no longer match.

For example, these are valid specifications:

RECOVER ... OBIDXLAT DBID(100,101) PSID (200,201)
INCOPY FULL DSNAME ... RBA X'123456'
or

```
RECOVER ... OBIDXLAT DBID(.101) PSID(.201)
INCOPY FULL DSNAMED ...
TOCOPY LASTCOPY
```

The following example is not valid:

```
RECOVER ... OBIDXLAT DBID(.101) PSID(.201)
INCOPY FULL DSNAMED ... RBA X'123456'
```

The syntax diagram for the OBIDXLAT specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**OBIDXLAT**

Use the OBIDXLAT keyword to change the internal IDs in the DB2 table space or index as the table space or index is recovered.

---

**Note**

Note the following information about the use of OBIDXLAT:

- When you use Instant Snapshots for migration and specify the OBIDXLAT option, NGT Recover automatically detects if the OBIDs have not changed and avoids the overhead of further processing.

- OBIDXLAT of Instant Snapshots of universal table spaces (UTS) further reduces the processing time, even if OBIDs change.

- If you specify RESET, NGT Recover does not get a speed improvement. For data sharing environments, RESET is usually not necessary. For non-data-sharing environments, if the RBAs on the target system are larger than the RBAs on the source, RESET is probably not necessary.

---

**RESET**

The RESET keyword causes the log points in each data page to be reset to '0'. You must specify RESET when you are migrating data from one DB2 non-data-sharing system to another because the log point values on the first subsystem is meaningless or misleading on the target DB2.

The recovery log point and level fields in the header page are also reset to 0 if you are not using the INDEPENDENT OUTSPACE option. If you are using the INDEPENDENT OUTSPACE option, the level fields are reset but the recovery log point is not.

```
DBID (X'hexSourceID', X'hexTargetID')
```
**DBID** *(decimalSourceID, decimalTargetID)*

Optionally, specify this clause to provide the DBIDs to be translated. You can specify the DBID in hexadecimal or decimal format. For more information, see “OBIDXLAT specification” on page 191.

**PSID** *(X'hexSourceID', X'hexTargetID')*  
**PSID** *(decimalSourceID, decimalTargetID)*

Optionally, specify this clause to provide the PSIDs to be translated. You can specify the PSID in hexadecimal or decimal format. For more information, see “OBIDXLAT specification” on page 191.

When you are recovering clone objects, remember that the base and clone objects are differentiated by differences in the PSID value. The high order bit of the PSID is used to refer to a particular data set instance number as follows:
- A high order bit value of 0 indicates instance number 1
- A high order bit value of 1 indicates instance number 2

Other than the high order bit, the PSID numbers are the same for both a base object and its clone. You should use the PSID with the high order bit set on for instance 2 objects in the OBIDXLAT translation specification.

**OBID** *(X'hexSourceID', X'hexTargetID')*  
**OBID** *(decimalSourceID, decimalTargetID)*

Optionally, specify this clause to provide the OBIDs to be translated. You can specify the OBID in hexadecimal or decimal format. For more information, see “OBIDXLAT specification” on page 191.

---

**OUTCOPY specification**

This topic describes the OUTCOPYDDN and RECOVERYDDN specifications that are included in the OUTCOPY specification.

The syntax diagram for the OUTCOPY specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

You can specify OUTCOPY to make copies of the object or object data set being recovered. For each object or data set that you process, you can request up to four image copies. The default is no copies. If you are recovering to the real DB2 data sets and you request image copies, you can choose which copies, if any, to register in the SYSIBM.SYSCOPY table (or BMCXCOPY table for indexes). However, if you are using the INDEPENDENT OUTSPACE option to recover to non-DB2 data sets and you request image copies, you cannot register those copies.

Output copies are produced in the new SYSTEMPAGES YES format when possible.
When you use the OUTCOPY ONLY facility, you can register the copies if you do not use OBIDXLAT or INCOPY or specify DROPRECOVERY.

If the output copies that you want to make are allocated in the JCL, the OUTCOPYDDN and RECOVERYDDN options specify the DD names for those data sets. If the output copies are dynamically allocated, the OUTCOPYDDN and RECOVERYDDN options refer to the appropriate OUTPUT descriptor names. For more information, see “OUTCOPYDDN specification” on page 198 and “RECOVERYDDN specification” on page 199. The copies that you want to register are specified by the REGISTER option (“INCOPY specification” on page 182).

When you create an OUTCOPY and want to force the copy to the cylinder-managed space of an extended address volume (EAV), specify the SPACE keyword.

**Note**

You can use NGT Recover to automatically size the OUTCOPY files generated by NGT Recover. For more information, see “Automatic sizing of output copies dynamically allocated on DASD” on page 499.

How the copies are made depends on whether the object is partitioned or nonpartitioned.

- If the object is partitioned, and OUTCOPY BYPART is specified or a specific DSNUM is coded in the object specification, a separate output copy is made for each partition. If you specified OUTCOPY ASCODED and DSNUM ALL with the RECOVER command, a single output copy is made for the object as a whole.

  For more information about using OPTIONS OUTCOPY BYPART, see “OUTCOPY” on page 116.

- If the object is nonpartitioned and you are recovering multiple data sets by specifying DSNUM ALL, each output copy is for all of the data sets. If you are recovering multiple data sets by specifying each one individually with DSNUM integer, a separate output copy is made for each data set.

**OUTCOPY NO**

OUTCOPY NO is the default and specifies that no output copies are required.

**OUTCOPY YES**

OUTCOPY YES specifies that you want image copies or DSN1COPY-type copies of the table space or index (or data set) being recovered. You can make up to four image copies or up to four DSN1COPY-type copies. A DD statement or an OUTPUT command statement must exist for each copy. The ddnames for the copy DD statements are specified by the OUTCOPYDDN and RECOVERYDDN options.
Note
You cannot use the OUTCOPY YES option to produce Instant Snapshot copies.

OUTCOPY ONLY

OUTCOPY ONLY does not affect the table space, index, or data set named in the RECOVER command statement and enables you to build copies without stopping the object for access and without actually reading the space, thereby avoiding DASD contention. You can use this option to build copies for a shadow version of the object, to build a migration image, or to build a registered copy for the named object’s recovery.

Note
If you use OUTCOPY ONLY to register a copy of a space at a point in time that is prior to existing incremental copies, the full copy will have a later TIMESTAMP, but a lower START_RBA than the associated incremental copies. Subsequent use of the IBM MODIFY utility to do an age-based DELETE from SYSCOPY may delete some of the incremental copies without deleting the associated full copy that was created by NGT Recover. The resulting gap in the incremental copies can cause errors in a subsequent recovery that uses the remaining incremental copies. BMC Software recommends using the NGT Copy MODIFY command to ensure that a consistent set of recovery resources is maintained in this situation.

The following restrictions apply to the OUTCOPY ONLY option:

- You cannot use the OUTCOPY ONLY option with the point-in-time recovery option BACKOUT.

- You cannot register an OUTCOPY ONLY copy to the same log point as an existing full copy for the same space or index. OUTCOPY ONLY copies cannot be registered if INCOPI, INLOG, or OBIDXLAT is specified except when DROPRECOVERY is also specified.

- You cannot generate a cabinet copy.

You can combine this option with an R+/CHANGE ACCUM product ACCUM command statement involving the same object, but only if the recovery does not include TORBA, TOLOGPOINT, or TOCOPY options. For more information about the ACCUM command statement, see the R+/CHANGE ACCUM for DB2 User Guide.

If copies are registered, they are registered as SHRLEVEL REFERENCE if any one of the following options is specified:

- TORBA or TOLOGPOINT to a quiesce point of the table space
- TORBA or TOLOGPOINT to a system wide quiesce point resulting from an ARCHIVE LOG MODE (QUIESCE) command

- TOCOPY to a SHRLEVEL REFERENCE incremental copy

- TORBA or TOLOGPOINT to a system shut down in a non-data-sharing environment

For all other cases, copies are registered as SHRLEVEL CHANGE.

**REGISTER**

Use this option to indicate how many, if any, copies you want to register in the SYSIBM.SYSCOPY table. Index copies are registered in the BMCXCOPY table.

You can make and register up to four copies, all with the same log point and the same SYSIBM.SYSCOPY time stamp. The first copy specified by ddname with the OUTCOPYDDN option is registered as the local site primary copy. Similarly, the second copy is registered as the local site backup copy. Third and fourth copies specified with RECOVERYDDN are registered as recovery site primary and backup copies.

**Note**

You cannot make a backup copy without a primary copy. You can, however, make and register a recovery site copy without making a local site copy.

**REGISTER ALL**

REGISTER ALL is the default and specifies that you want to register all of the copies that NGT Recover makes for the object in the SYSIBM.SYSCOPY or BMCXCOPY table.

**REGISTER NONE**

If you specify OUTCOPY YES REGISTER NONE, NGT Recover makes up to four DSN1COPY-type copies.

**REGISTER (DDNameList)**

OUTCOPY YES REGISTER *(DDNameList)* specifies by ddname those image copies that you want to register in the SYSIBM.SYSCOPY or BMCXCOPY table. The ddname list is optionally enclosed in parentheses and that the ddnames must be separated by commas. For each name in the list, there must be a ddname, ddname prefix, or output descriptor specified in OUTCOPYDDN/RECOVERYDDN explicitly or implicitly as the default value.
The entire OUTCOPYDDN/RECOVERYDDN specification can be omitted and NGT Recover will use the default ddnames. Also note, REGISTER allows the use of any form of the defaults (BMCCPY or BMCCY, BMCCPZ or BMCCZ, and so on) to refer to the default ddnames. Both forms have the same meaning for the REGISTER specification if the corresponding ddname in OUTCOPYDDN/RECOVERYDDN is defaulted. For more information “RECOVERYDDN specification” on page 199 and “Copy data set ddname construction for JCL-allocated data sets” on page 214.

### RESETRTS

Use RESETRTS to have NGT Recover reset the copy DB2 real-time statistics (RTS) if the copy is registered when you specify OUTCOPY YES or OUTCOPY ONLY.

### WARNING

You should specify RESETRTS only if the OUTCOPY you are making is a copy that approximately represents the current state of the space. If you specify RESETRTS for a copy corresponding to a different point in time, the RTS may give misleading information about the time of the last copy and the changes made since the copy.

Several examples of how you might use RESETRTS follow:

- You might use RESETRTS if you want to make a copy of your system at a specified point in time, such as at midnight. Because making GROUP YES SHRLEVEL REFERENCE copies of the whole system is not usually practical, another approach is to issue an ARCHIVE LOG MODE(QUIESCE) command to get a system-wide point of consistency. You can then use NGT Recover OUTCOPY to make copies of all of the spaces at the point of consistency. While NGT Recover makes copies with OUTCOPY, the system is available for update. Using this method, you get consistent copies of the whole system at the desired point in time with minimal outage. This approach is sometimes referred to as the stealth copy procedure. You should know when you use this procedure that the copy is very close to the current state of the system, and you should use RESETRTS to reset the RTS copy statistics.

- You might want to use RTS to trigger utility runs. For example, you might want to make a copy if RTS shows that the last copy was made more than 2 days ago. When you use NGT Recover to recover a space and make an OUTCOPY, NGT Recover resets the RTS copy statistics and updates the RTS timestamp for the last copy. However, if you run NGT Recover OUTCOPY ONLY and do not specify RESETRTS, NGT Recover does not reset RTS because the space was not updated and the output copy does not necessarily match the space. For example, the current date and time

---

Chapter 3  NGT Recover syntax  197
might be 10 AM on October 15, but the OUTCOPY might be for 10 PM on October 12.

**TAG tagName**

Specify TAG to tag copies with a name that can be used for selecting objects for use with the NGT Copy EXPORT command. Only image copies for table spaces are tagged.

Tag names are presented as 1 to 32 characters, followed by a generation number in parentheses. The generation number can be (0), (+1), or (-n). Generation numbers are used as follows:

- (0)—uses the highest generation of the tag.
- (+1)—uses the highest generation of the tag, plus 1.
- (-n)—uses the highest generation of the tag, less n generations.

Spaces and punctuation marks are not allowed in the tag name. Specify the generation number as in the following example: WEEKLY_EXPORT(+1).

**SPACE (primary, secondary) allocation unit**

Specify SPACE to set the output allocation parameters for the named descriptor in OUTCOPYDDN. This use of SPACE overrides the SPACE specification in the corresponding OUTPUT descriptor. Specify SPACE TRK to allocate the output in tracks. Specify SPACE CYL to allocate the output in cylinders. The default value for this option is CYL. Specify the values in parentheses, as in the following example: SPACE (200,100) CYL. If no SPACE specification is provided, the default is (5,5) CYL.

**OUTCOPYDDN specification**

OUTCOPYDDN overrides the default ddnames or ddname prefixes that are used by NGT Recover.

The syntax diagram for the OUTCOPYDDN specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**OUTCOPYDDN (DDName1,DDName2)**

You can use OUTCOPYDDN (DDName1,DDName2) to override the default ddnames or ddname prefixes that are used by NGT Recover for a table space or index to specify copy data set ddnames in the JCL. OUTCOPYDDN can also reference an output descriptor named with an OUTPUT command statement if the copy data set is to be dynamically allocated.
The following restrictions apply only if OUTCOPYDDN refers to a DD statement in the JCL:

- When a table space or index has fewer than 100 partitions, you can specify any value of up to six characters. If copies are being made by partition, this value is a prefix. If DSNUM ALL and OPTIONS OUTCOPY ASCODED are specified, this value is a ddname.

- For a table space or index with 100 or more partitions, the default values for all partitions are BMCCY and BMCCZ, and you can specify any value of no more than five characters.

- For a nonpartitioned table space or index, if DSNUM ALL is specified or the highest value of DSNUM in the run is less than 100, you can specify any value up to six characters. If the highest value of DSNUM is 100 or higher, you can specify any value of up to five characters.

The ddnames specified with OUTCOPYDDN are optionally enclosed in parentheses but must be separated by a comma. For more information, see “Copy data set ddname construction for JCL-allocated data sets” on page 214.

**OUTCOPYDSN** *(dataSetName1, dataSetName2)*

You can use OUTCOPYDSN *(dataSetName1, dataSetName2)* to override the data set name specified by an output descriptor for OUTCOPYDDN for a dynamically allocated output image copy. Use of OUTCOPYDSN when the output image copy is not dynamically allocated is an error.

**RECOVERYDDN specification**

RECOVERYDDN overrides the default ddnames or ddname prefixes that are used by NGT Recover.

The syntax diagram for the RECOVERYDDN specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**RECOVERYDDN** *(DDName3,DDName4)*

You can use RECOVERYDDN *(DDName3,DDName4)* to override the default ddnames or ddname prefixes that are used by NGT Recover for a table space or index to specify copy data set ddnames in the JCL. RECOVERYDDN can also reference an output descriptor named with an OUTPUT command statement if the copy data set is to be dynamically allocated.
The following restrictions apply only if RECOVERYDDN refers to a DD statement in the JCL:

■ When a table space or index has fewer than 100 partitions, you can specify any value of up to 6 characters. If copies are being made by partition, this value is a prefix. If DSNUM ALL and OPTION OUTCOPY ASCODED are specified, this value is a ddname.

■ For a table space or index with 100 or more partitions, the default values for all partitions are BMCRY and BMCRZ, and you can specify any value of no more than 5 characters.

■ For a nonpartitioned table space or index, if DSNUM ALL is specified or the highest value of DSNUM in the run is less than 100, you can specify any value up to six characters. If the highest value of DSNUM is 100 or higher, you can specify any value of up to five characters.

The ddnames specified with RECOVERYDDN are optionally enclosed in parentheses but must be separated by a comma. For more information, see Copy data set ddname construction for JCL-allocated data sets on page 214.

**RECOVERYDSN (dataSetName3, dataSetName4)**

You can use RECOVERYDSN (dataSetName3, dataSetName4) to override the data set name specified by an output descriptor for RECOVERYDDN for a dynamically allocated output image copy. Use of RECOVERYDSN when the output image copy is not dynamically allocated is an error.

**SCOPE specification**

Use the SCOPE options to enable NGT Recover to bypass objects based on specified criteria, thereby defining the scope of the recovery.

To use SCOPE to reduce the scope of an object list, you must first supply the object list. Therefore, RECOVER TABLESPACE SCOPE STATUS (RECP) is not supported.

The SCOPE option is not allowed if NGT Recover is processing clones, OUTCOPY is requested and OUTCOPY BYPART is not in use, or you are using any of the following options:

■ RICHK YES
■ IXRECP YES
■ NOSYSLGRNG
■ OUTCOPY ONLY
■ CLONE

NGT Recover ignores SCOPE UPDATED if it is specified for recovery to current.
NGT Recover ignores SCOPE UPDATED if it is specified for recovery to current or for a backout recovery, or if any of the following options is specified:

- DROPRECOVERY
- INCOPY
- INDEP OUTSPACE
- LOGAPPLY ONLY
- LOGONLY

For SCOPE syntax, see “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**SCOPE UPDATED**

To force a point-in-time recovery at the local site when a space has not changed since the recovery point, specify OPTION RECOVERSCOPE ALL. NGT Recover bypasses spaces that have not changed since a specified recovery point.

SCOPE UPDATED is the default option for local site point-in-time forward recovery requests when OPTION RECOVERSCOPE is not specified.

```
Example
RECOVER TABLESPACE OBJECTSET <oscreator>.<osname>
    TOLOGPOINT LASTQUIESCE
    SCOPE UPDATED
```

**SCOPE ALL**

SCOPE ALL is the default option for all recoveries to current and remote site recoveries. If you specify SCOPE ALL, NGT Recover recovers all specified spaces.

**SCOPE STATUS(status1, status2,...)**

NGT Recover bypasses spaces that are not in at least one of the specified statuses.

```
Example
RECOVER TABLESPACE OBJECTSET <oscreator>.<osname>
    SCOPE STATUS(GRECP,LPL)
```

**SCOPE PENDING**

NGT Recover bypasses spaces that are not in RBDP, RBDP*, PSRBD, or RECP status.

To process spaces with other statuses, use SCOPE STATUS(...).
TOCOPY specification

TOCOPY specifies the last image copy to be processed.

For TOCOPY syntax, see “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

TOCOPY

The TOCOPY option specifies the last image copy to be processed by NGT Recover to recover the specified table space (full or incremental copies) or index (full copies only). The recovery process ends after this image copy has been applied. If you specify an incremental copy for table space recovery, the prior full copy and any intervening incremental copies are also used in the restore process. Except when used with the INCOPY option, the image copy specified must be registered in SYSIBM.SYSCOPY (or in BMCXCOPY for indexes) as an image copy or as a "kept" incremental copy (as specified by the BMC Next Generation Technology Copy for DB2 for z/OS product) for the specified object or object data set. When you specify TOCOPY, the log is not used to apply updates that are made after the copy.

After a TOCOPY recovery, the data in the space reflects the data as of the time of the specified image copy. Any log record or activity after the TOCOPY recovery is no longer reflected in the space but may be reinstated by a subsequent TOLOGPOINT recovery.

You can specify the image copy by its data set name or by the keyword LASTCOPY.

If the copy named was created by using SHRLEVEL CHANGE, NGT Recover issues a warning (message BMC40971), but continues processing. (With a SHRLEVEL CHANGE copy, the pages may not be logically consistent.)

Note

When you use TOCOPY, you may specify only one table space. To recover more than one table space by using TOCOPY, you can use multiple RECOVER TABLESPACE command statements. The same is true for index recovery with RECOVER INDEX or RECOVER INDEXSPACE.

If you are recovering directly to the real DB2 data sets (not using INDEPENDENT OUTSPACE and not using OUTCOPY ONLY), a point-in-time recovery event is recorded in SYSIBM.SYSCOPY or BMCXCOPY. However, if you are not changing the actual objects and specify TOCOPY, the event is not recorded in
SYSIBM.SYSCOPY (or BMCXCOPY) because the actual DB2 data sets are not changed.

**TOCOPY LASTCOPY** *(relativeGenerationNumber)*

TOCOPY LASTCOPY specifies that the most recent image copy (full or incremental for table spaces, full for indexes) registered in SYSIBM.SYSCOPY (or BMCXCOPY for indexes) is the last one to be processed by NGT Recover to recover the specified object.

 Optionally, you can add a relative generation number in parentheses to indicate the copy to use. A relative generation number of (0) uses the most recent image copy. For example, TOCOPY LASTCOPY (-1) indicates the use of the copy before the most recent copy.

When used with the INCOPY option, LASTCOPY means the last copy specified in your INCOPY list.

**TOCOPY dataSetName**

TOCOPY dataSetName specifies that the image copy data set named (full or incremental for table spaces, full for indexes) is the last one to be processed by NGT Recover to recover the object. The named copy must be registered in SYSIBM.SYSCOPY (or BMCXCOPY for indexes or Instant Snapshot copies) unless you are choosing the INCOPY option. In that case the named copy must be the last copy specified in your INCOPY list. When the data set is part of a generation data group (GDG), you can specify either the full name or the relative GDG name.

If you specify a relative GDG name for an existing data set, use either (0) or (-n) for the relative generation. Do not refer to it as the (+1) generation. The most recent data set is the (0) generation, even if it was created in a previous step in the same job.

If the image copy is an Instant Snapshot copy, you should specify the VSAM data component name or the cluster component name. (The VSAM data component name is printed in the output from the NGT Copy job that created the Instant Snapshot.) In reporting, NGT Recover always shows only the data component name.

The following restrictions apply to the TOCOPY dataSetName option:

- You cannot specify a data set named LASTCOPY when you specify TOCOPY data set names.

- If dataSetName is not cataloged, you must use the TOVOLUME option to complete the identification. TOVOLUME identifies the volume containing the image copy data set to be used for recovery. You can use TOVOLUME only with TOCOPY dataSetName. The value that you specify with
TOVOLUME depends on whether the data set is cataloged or uncataloged.

**Note**

TOCOPY *dataSetName* is not allowed with OBJECTSET.

**TOVOLUME CATALOG**

TOVOLUME CATALOG is the default and identifies the image copy data set as a cataloged data set. It specifies that the volume serial number should be obtained from the operating system catalog.

**TOVOLUME volumeSerialNumber**

TOVOLUME *volumeSerialNumber* identifies the image copy data set as uncataloged, where *volumeSerialNumber* is the serial number of the first volume containing the image copy data set. (If the copy is in multiple volumes, the serial number of the first volume is the only one you need to specify. NGT Recover determines all other volumes from SYSCOPY or BMCXCOPY.)

**TOSEQNO integer**

The *integer* parameter for TOSEQNO is the sequence number of the data set containing the image copy data set on the tape volume. You can use TOSEQNO only when you specify both TOCOPY and TOVOLUME *volumeSerialNumber*. If you specify TOCOPY and TOVOLUME *volumeSerialNumber* (to indicate an uncataloged data set) but do not specify TOSEQNO, NGT Recover assumes a value of 1 if the copy is cataloged. Otherwise, NGT Recover searches SYSCOPY or BMCXCOPY for a matching volume and uses the sequence number found.

**TORBA or TOLOGPOINT specification**

Use TORBA or TOLOGPOINT to recover a table space or index to a prior point in time identified by a log point.

The syntax diagram for the TORBA or TOLOGPOINT specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

**Note**

Place syntax for the TORBA or TOLOGPOINT specification after the syntax for Table space specification, Single index specification, or Multiple index specification.
For more information about using TORBA or TOLOGPOINT with table spaces and indexes, see “Additional RECOVER INDEX and RECOVER INDEXSPACE limitations” on page 165.

**TORBA or TOLOGPOINT**

You can use TORBA or TOLOGPOINT interchangeably, regardless of the version of DB2 that you are using.

A TOLOGPOINT recovery returns a space to a point that reflects all activity on the space up to and including a log record at the log point specified. Any log record or activity higher than TOLOGPOINT is no longer reflected in the space but may be reinstated by a subsequent TOLOGPOINT recovery.

---

**Note**

If you are recovering directly to the real DB2 data sets and are not using INDEPENDENT OUTSPACE, a point-in-time recovery event is recorded in SYSIBM.SYSCOPY for table spaces and COPY YES indexes or BMCXCOPY for COPY NO indexes.

In contrast, if you are using the INDEPENDENT OUTSPACE option to recover to non-DB2 data sets and specify TORBA or TOLOGPOINT, the event is not recorded in SYSIBM.SYSCOPY or BMCXCOPY because the actual DB2 data sets are not changed. Using OUTCOPY ONLY with TORBA or TOLOGPOINT is not a point-in-time recovery; instead, you are simply creation of a copy as of an earlier point in time. You can register the copy in SYSCOPY or BMCXCOPY, but no other event is recorded.

---

**TORBA LASTQUIESCE**

You can use TORBA LASTQUIESCE or TOLOGPOINT LASTQUIESCE to recover to the most recent quiesce point registered in SYSIBM.SYSCOPY for the table space being recovered. For index recovery, the most recent quiesce of the table space is used.

Optionally, you can add a relative generation number in parentheses to indicate the quiesce to use. A relative generation number of (0) uses the most recent image copy. For example, TOLOGPOINT LASTQUIESCE (-1) indicates the use of the quiesce before the most recent quiesce.

If you are recovering multiple table spaces in a single RECOVER TABLESPACE command statement, all of the table spaces must have a common last quiesce point. However, if you are using multiple RECOVER TABLESPACE command statements to recover multiple table spaces, each table space can have a different last quiesce point. The same is true for index recovery with RECOVER INDEX or RECOVER INDEXSPACE.

LASTQUIESCE can be abbreviated as LASTQ.
**Note**

If you use LASTQUIESCE and the quiesce is prior to an ALTER ADD PART, the recovery will fail. You must specify a hard-coded RBA. If the partition was added for a partition-by-growth universal table space, you can use the LASTQUIESCE option.

---

**TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ**

Use TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ to recover a set of table spaces and indexes to the most recent common quiesce point registered in SYSIBM.SYSCOPY. Index recoveries are based on the common quiesce point of the owning table space. If no common quiesce point exists for all table spaces and indexes specified, an error message is issued.

If you use this option when only one table space or index is recovered, the effect is exactly the same as using TORBA LASTQUIESCE.

You can use TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ with RECOVER INDEX (ALL) or when multiple index space specifications exists in one command statement.

**TORBA LASTARCHQ or TOLOGPOINT LASTARCHQ**

Use TORBA LASTARCHQ or TOLOGPOINT LASTARCHQ to recover to the log point of the last ARCHIVE LOG MODE(QUIESCE) command issued for the DB2 subsystem.

When you use this option, NGT Recover issues a message indicating the date and time of the ARCHIVE LOG MODE(QUIESCE) command used.

**TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN**

Use TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN to recover to the log point of the last normal system shutdown. When you use this option, NGT Recover issues a message indicating the date and time of the last successful system shutdown. You cannot use the LASTSHUTDOWN keyword to recover to the log point of an abnormal DB2 termination. When you use this keyword, ensure that the table spaces and indexes were not in an exception status at the shutdown.
In a data sharing environment, this keyword is used to recover to the last successful shutdown of the member subsystem on which the job is running, without regard to other member subsystems. For this reason, use extreme caution when you use TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN in a data sharing environment to ensure that the shutdown used is really a point of data consistency.

To ensure that the last successful shutdown is really a point of data consistency, one of the following conditions must be true:

- The subsystem on which the NGT Recover job is running was the last member of a data sharing group to be stopped.
- All updates to the table spaces and indexes to be recovered occurred on the subsystem on which the NGT Recover job is running.

**TORBA LOGMARK** logMarkName (logMarkGeneration) or **TOLOGPOINT LOGMARK** logMarkName (logMarkGeneration)

Use TORBA LOGMARK logMarkName or TOLOGPOINT LOGMARK logMarkName to recover to a prior point in time identified by a log mark that was created in Log Master.

Optionally, you can add a log mark generation number in parentheses, (logMarkGeneration). If you do not specify a log mark generation, NGT Recover uses the most recent version of the log mark. You can use one of the following ways to specify the log mark generation:

- If you specify a log mark generation as less than or equal to zero (\(< = 0\) ), NGT Recover treats the generation as a relative generation. NGT Recover refers to the most recent log mark with the generation of zero (0). The previous generation of the log mark is referred to with (-1), and so on.
- If you specify a log mark generation as greater than zero (\(> 0\) ), NGT Recover treats the generation as an absolute generation number and uses the specified version of the log mark.

**Note**

The way that you specify an absolute generation number for a log mark in NGT Recover is different than the way you specify the absolute generation number in Log Master.

When you create the log mark, you can specify if Log Master should create a quiesce point. You can also specify that Log Master set a log mark at a quiet point or a non-quiet point. If you know that the log mark is at a quiesce or a quiet point for the space or spaces that you are recovering, use this option to perform a point-in-time recovery. If the log mark is not at a quiet point, you
should use a timestamp recovery using option RECOVERYPOINT
LOGMARK logMarkName.

Note
You must determine if the log mark is at a quiesce or quiet point or not. If
you are not sure that there is a quiesce or quiet point, a timestamp recovery
may be best although it will take longer to look for inflight transactions.

TORBA X'logPoint' or TOLOGPOINT X'logPoint'

Use TORBA X'logPoint' or TOLOGPOINT X'logPoint' to recover to a prior
point in time identified by the log point, 'logPoint'. Except with BACKOUT,
only log records with starting log points less than or equal to 'logPoint' are
used by NGT Recover.

'logPoint' is a string of up to twelve hexadecimal digits.

BACKOUT

BACKOUT invokes the backout strategy for point-in-time recovery by
using log points (TORBA, TOLOGPOINT). This strategy assumes that
spaces are undamaged and that you require a reset to a specific point in
time. The spaces are used with the log records between the point in time
and the current point to back out to the required state. Using BACKOUT
may enhance the performance of a point-in-time recovery significantly.

The following restrictions apply:

- All recovery requests in SYSIN must use BACKOUT. You cannot mix
  backward and forward recovery in SYSIN.

- The space must be current as of the last logged activity and not
damaged in any way. Multi-data-set, nonpartitioned spaces must have
all data sets recovered (DSNUM ALL).

- The space must not be in RECP, RECP*, RBDP, RBDP*, PSRCP, PSRBD,
GRECP, WEPR, REFP, or STOPE status or have an LPL range.

- No LOAD or REORG events can exist between the log point specified
and the current point in time. For indexes, no REBUILD INDEX events
can exist in this range.

- No prior point-in-time recovery with a START_RBA greater than the
log point requested and a PIT_RBA less than the log point requested
can exist.

- BACKOUT may not be requested with the following NGT Recover
options:
  — INDEP OUTSPACE
— OUTCOPY ONLY
— OBIDXLAT
— DROPRECOVERY
— LOGONLY or LOGAPPLY ONLY

**Note**
BACKOUT uses only logs and spaces but requires that the spaces be current. The LOGONLY and LOGAPPLY ONLY options imply applying log records going forward by using a space restored to a previous state.

- Change accumulation files are not allowed with BACKOUT because they are not properly ordered. Output accumulation files are also not supported because they are defined from the point of the last image copy to the current point in time.

**ANALYZE**

The ANALYZE option prints an NGT Recover plan before executing that plan.

The following information about objects is included in the plan:

- Names of any image copy data sets on which activities are or will be based
- Names of any log data sets on which activities are or will be based
- Name of any change accumulation files on which activities are or will be based
- Log ranges, if any, on which activities are or will be based
- Number of log pages to be read
- Record sizes for index sort work data sets
- Phases to occur during execution
- Steps to occur within each phase

Use this information to allocate space more accurately for log sort work data sets, index sort work data sets, and index key work data sets, thereby limiting abends that are caused by inadequate data set allocations. You can use these statistics with historical information about the times required to perform the various operations to estimate recovery time.

**Note**
If you coded ANALYZE with the OPTIONS command, you must code the same value here. If you code multiple NGT Recover command statements you must use the same value for the ANALYZE option in all of the command statements. If you specify a value for ANALYZE on one command statement and take the default on the others, the value that you specified on the single command statement becomes the default for the others.
ANALYZE YES

ANALYZE YES is the default and provides the information listed in the preceding section.

ANALYZE NO

If you specify ANALYZE NO, NGT Recover does not provide any plan information, but does provide object summaries.

ANALYZE ONLY

If you specify ANALYZE ONLY, NGT Recover provides the same information as it provides with ANALYZE YES and then stops the job when the ANALYZE phase is complete. You can use the data from the ANALYZE phase to determine the resources required and what will happen during the run. You cannot restart an NGT Recover job that specifies ANALYZE ONLY, but you can start a new job.

Use the ANALYZE ONLY option to determine which phases will occur with a specific request, or which copy data sets and log data sets will be used.

If you provide a SYSPICK DD statement, NGT Recover generates a list of all of the input tape and cartridge volumes that are allocated. For more information about SYSPICK, see “NGT Recover data sets and NGT Recover DD statements” on page 319.

If you specify OPTIONS EARLYRECALL with ANALYZE ONLY, the recall of the data sets needed for execution will be initiated. You must code OPTIONS EARLYRECALL explicitly to initiate the recalls, because OPTIONS NOEARLYRECALL is the default for ANALYZE ONLY executions.

CLONE

The CLONE option indicates that NGT Recover is to recover clone index data. When you work with clone data, the following limitations apply:

- You cannot refer to the same object with the CLONE option and without the CLONE option in the same recovery step; you cannot process the base and its clone in the same command.

- Related objects (table space and all indexes for its table, all related LOB objects) should use the same CLONE specification in the same recovery step.

DEFINE

DEFINE specifies whether DEFINE NO objects will be instantiated.
**Note**

IBM DB2 Version 11 or later supports the DEFINE option.

---

**DEFINE YES**

NGT Recover instantiates DEFINE NO objects, if necessary, and updates the catalog and DBD to reflect that the object has been defined.

**DEFINE NO**

NGT Recover does not process DEFINE NO objects. NO is the default value.

**LOCALSITE**

LOCALSITE indicates that only the image copies created for the local site are used in the recovery. The image copies are used in the order specified in the installation options module for local site:

```plaintext
BMC96111I DEFAULT RESOURCE SELECTION SEQUENCE FOR LOCAL SITE
BMC96113I COPIES = (LP,LB) (DEFAULT SEQUENCE FOR IMAGE COPIES)
```

---

**RECOVERYSITE**

RECOVERYSITE indicates that only the image copies for the recovery site are used in the recovery. The image copies are used in the order specified in the installation options module for remote site:

```plaintext
BMC96112I DEFAULT RESOURCE SELECTION SEQUENCE FOR REMOTE SITE
BMC96113I COPIES = (RP,RB) (DEFAULT SEQUENCE FOR IMAGE COPIES)
```

---

**LOGAPPLY ONLY**

LOGAPPLY ONLY specifies that only log information is to be used to update a table space or index and that no image copy data sets are to be used. Use this option when you want to specify a FROMRBA or FROMLOGPOINT.

LOGAPPLY ONLY is useful when another job has already recovered one or more table spaces or indexes to a known state. LOGAPPLY ONLY directs NGT Recover to avoid merging and restoring copies and to apply only log updates from the FROMRBA or FROMLOGPOINT specified.

When you want to use only log records to update multiple table spaces or indexes in a single RECOVER TABLESPACE, RECOVER INDEX, or RECOVER INDEXSPACE command statement, you need to specify only a single LOGAPPLY ONLY. However, you must specify FROMRBA or FROMLOGPOINT for each table space or index.
When you use LOGAPPLY ONLY, you cannot use the TOCOPY option, the INCOPY option, or the INLOG option.

**LOGONLY**

LOGONLY applies only the log records to the data sets. All log records written after a point recorded in the data set are applied. Specify LOGONLY when the data sets of the target objects have been restored by using another process and the HPGRBRBA value in the header pages are correctly set to indicate where log apply should begin. You can use this option on single table spaces or indexes, or on table space or index lists.

When you use LOGONLY, you cannot use the TOCOPY option, the INCOPY option, or the INLOG option.

**REDEFINE**

Use the REDEFINE option to avoid the normal deletion and reallocation of the object data sets for STOGROUP-defined objects. This option is ignored for VCAT-defined object data sets. The value that you specify for REDEFINE applies to all STOGROUP-defined objects included with a single NGT Recover command. To recover multiple STOGROUP-defined objects with some reallocated and others not reallocated, you must use one NGT Recover command statement for the objects to be reallocated and a second NGT Recover command statement for those objects not to be reallocated.

Regardless of the value of REDEFINE, if the data sets for a STOGROUP-defined object do not exist, NGT Recover creates the VSAM data sets.

---

**Note**

When a point-in-time recovery of a multi-data-set, nonpartitioned object is performed, more or fewer data sets may result than were previously used. If data sets were added after the point-in-time recovery point, there are fewer data sets after the point-in-time recovery. If the object was reorganized after the point-in-time recovery point, there may be more data sets.

---

**REDEFINE YES**

REDEFINE YES is the default. If you specify REDEFINE YES, NGT Recover deletes and reallocates the VSAM data sets for STOGROUP-defined objects prior to the recovery. For more information, see Storage group-defined data sets on page 80.

**NOSCRATCH**

If you specify NOSCRATCH with REDEFINE YES, NGT Recover invokes IDCAMS to uncatalog without scratching. You can use NOSCRATCH to support disaster recovery scenarios where the following conditions are true:
DASD volumes at the recovery site differ from those at the primary site.

The operating system catalog and the DB2 catalog and directory are restored before running NGT Recover.

DB2 STOGROUPS are changed to point to the recovery site packs.

Using this option eliminates mount messages for the primary site packs that do not exist at the recovery site.

**Note**

REDEFINE YES has no effect if you specify INDEP OUTSPACE. For the instructions for creating data sets for INDEP OUTSPACE, see “INDEPENDENT OUTSPACE specification” on page 185. If an Instant Snapshot copy is used to recover the object, REDEFINE YES has no effect and the data sets are processed as if REDEFINE NO had been specified.

**REDEFINE NO**

REDEFINE NO tells NGT Recover not to delete and reallocate the VSAM data sets for the object. This option enables you to treat STOGROUP-defined objects as VCAT-defined objects and to reallocate the VSAM data sets manually, or use the data sets already assigned to the object. New volumes are used as needed from the STOGROUP, even with REDEFINE NO specified.

**Note**

With REDEFINE NO specified, when recovery of a multi-data-set, nonpartitioned object is performed as DSNUM ALL, any necessary additional data sets are created and unused data sets are deleted.

When you specify REDEFINE NO, NGT Recover does not free unused extents in a retained data set. If you want the recovery to free unused extents, specify REDEFINE YES, which is the default.

**REUSE**

The REUSE option specifies that the space will not be deleted or redefined. This option cannot be used with the REDEFINE option.

**UPDATE VERSIONS**

Use the UPDATE VERSIONS option to have NGT Recover call the DB2 REPAIR VERSIONS utility after a table space recovery of a versioned table space is complete. The DB2 REPAIR VERSIONS utility updates the version information in the DB2 catalog and directory so that the source and target information match.
The UPDATE VERSIONS option is useful if you migrate data from one DB2 system to another using NGT Recover. Using NGT Recover is a fast way to restore a copy from the source system to the target system.

If you use NGT Recover and this option to perform migration, you need to:

- Make sure that the input copy has system pages for all versions of the data
  One way to do this is to use NGT Copy with the GENSYSPAGES option to make the copy.
- Recover table spaces in a separate step from rebuilding indexes

**Copy data set ddname construction for JCL-allocated data sets**

The copy data set ddnames specified with the REGISTER, OUTCOPYDDN, and RECOVERYDDN options must be coded as follows in the JCL:

- If any of the following conditions exist, the ddnames specified with the REGISTER, OUTCOPYDDN, and RECOVERYDDN options (or the defaults) are coded in the JCL as prefixes to the partition or data set numbers:
  - The object specification is for a specific data set or partition (DSNUM integer specified).
  - The object is partitioned and OUTCOPY BYPART is coded with the OPTIONS command.
  - The object is partitioned and the OUTCOPY installation option value is BYPART.

In the JCL, the ddnames must be in the form namenn or namennn, where name is the ddname specified with REGISTER, OUTCOPYDDN, or RECOVERYDDN and cannot be more than six characters for objects with less than 100 partitions or five characters for objects with greater than 99 partitions.

For a partitioned table space or index, nnn is the partition number and must be in the range 1 through 999.

**Note**

For partitioned spaces with more than 999 partitions, you must use dynamic allocation. You can have 4096 dynamically allocated data sets.

For a nonpartitioned table space or index, nn is the number of a single data set and must be in the range 1 through 32 for table spaces, or 1 through 128 for indexes.
If the object specification is for all of the data sets of a nonpartitioned object (DSNUM ALL specified) or for all partitions of a partitioned object, and the OUTCOPY option is ASCODED, then the ddnames specified with the REGISTER, OUTCOPYDDN, and RECOVERYDDN options are used "as is" in the JCL. When a request of this type is made, the copy made is for all of the data sets of the object. The ddname cannot be more than eight characters.

Using the specified ddnames in this way ensures that the output copy for each object or partition has a unique ddname. Output copies cannot share data sets, although output copies for multiple object partitions can share the same prefix because the partition number suffix makes each copy DD statement unique.

REBUILD INDEX

This topic describes the syntax of the REBUILD INDEX command and its options.

Considerations

- When INDEXLOG is set to NO in the installation options or with the OPTIONS command, RECOVER INDEX and REBUILD INDEX are synonyms. Any reference to REBUILD INDEX in this book also applies to RECOVER INDEX when INDEXLOG is set to NO. RECOVER INDEXSPACE, however, is never a synonym for REBUILD INDEX, regardless of the value of INDEXLOG.

- When an index is rebuilt, key values are instantiated in the current version of the table, regardless of the version of the table row from which the key value was extracted. (DB2 added online schema evolution that provided versioning of tables.)

- When you use REBUILD INDEX with a RECOVER TABLESPACE that includes OBIDXLAT and INDEP OUTSPACE, the OBIDXLAT in the RECOVER TABLESPACE statement must specify the OBIDs for the indexes specified in the REBUILD INDEX statement. Because REBUILD INDEX does not have an OBIDXLAT clause, specifying the OBIDs in the RECOVER TABLESPACE statement is the only way to get the OBIDs translated in the index header page. For more information, see OBIDXLAT specification on page 191.
REBUILD INDEX syntax

The following figures show the syntax of the REBUILD INDEX command.

Figure 30: REBUILD INDEX syntax diagram

1 Is synonymous with REBUILD when INDEXLOG=NO
2 Is synonymous with SIMRBLD when INDEXLOG=NO
Figure 31: REBUILD INDEX syntax diagram

Figure 32: REBUILD INDEX syntax diagram—OBJECTSET specification

OBJECTSET specification

Chapter 3  NGT Recover syntax   217
**REBUILD INDEX option descriptions**

NGT Recover provides the following options for use with the REBUILD INDEX command. The options are described in the order in which they are shown in the REBUILD INDEX syntax diagram.

**REBUILD**

Use the REBUILD command when you want to rebuild an index from data.

**RECOVER**

The RECOVER command is synonymous with the REBUILD command when INDEXLOG is set to NO.
When you simulate recovery by specifying the SIMULATE YES option, you should use SIMRCVR rather than RECOVER. For more information, see “SIMRCVR” on page 167.

SIMRBLD

SIMRBLD is a synonym for REBUILD. Use the SIMRBLD command to clarify that you are running in simulation mode.

SIMRBLD command generally has the same meaning and the same syntax in simulation mode as a regular REBUILD command has in non-simulation mode.

When you use SIMRBLD, you must also specify OPTIONS SIMULATE YES, which requires a valid BMC Recovery Management for DB2 solution password. For more information, see “SIMULATE YES” on page 132.

SIMRCVR

SIMRCVR is a synonym for SIMRBLD when INDEXLOG is set to NO.

For more information, see “SIMRBLD” on page 219.

INDEX

The INDEX option specifies the indexes (and the corresponding index spaces) to be rebuilt.

You can specify one or more indexes in one command statement.

Note

NGT Recover does not allow the rebuilding of indexes into DB2 data sets if the corresponding table space recovery is to a non-DB2 (independent) data set or if the corresponding table space recovery is a point-in-time recovery OUTCOPY ONLY.

(indexName,indexName2, . . . . . .)

This parameter specifies each of the indexes that you want to rebuild. Each index name is in the form authid.indexName. If you do not provide the authid qualifier, the qualifier defaults to the user ID of the user executing the utility. The list of qualified names must be enclosed in parentheses and separated by commas. You cannot specify any of the indexes on DB2 tables that are used by NGT Recover (such as those on BMCSYNC).
PART integer

This option specifies the partition of the index that you want to rebuild.

*Note*

Rebuilding a logical partition is not supported. You must rebuild the entire nonpartitioned index.

PART begin : end

Use this option to recover a range of partitions. For partitioned spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

ALL

INDEX(ALL) specifies that all indexes are to be rebuilt for the table space named in the TABLESPACE option.

OBJECTSET specification

Use OBJECTSET (REBUILD INDEX OBJECTSET creator.name) to specify that you want to rebuild all of the indexes in a group that was defined using RECOVERY MANAGER.

*Note*

SIMRBLD is also valid in place of REBUILD.

The OBJECTSET specification is shown in “REBUILD INDEX syntax” on page 216.

OBJECTSET creator.name

The following rules apply to creator.name:

- creator specifies the name of the creator of the group and can have up to 128 characters in length.
- name specifies the group name and can have up to 128 characters in length.
- You can delimit both creator and name with single or double quotation marks.
- Both creator and name can contain the special characters $, #, and @ in any position.

NGT Recover uses dynamic grouping to determine the group contents at the time that you run the NGT Recover job containing the OBJECTSET option. Because NGT
Recover reads the objects in the group each time the job is executed, objects may be added or removed from the group.

**Note**

NGT Recover does not read RECOVERY MANAGER recovery options for the group. If RECOVERY MANAGER recovery options change, you must run RECOVERY MANAGER to pick up the new options and generate the control cards.

Following are restrictions that apply when you use OBJECTSET with REBUILD INDEX:

- AUX is ignored with any specification of OBJECTSET.
- You cannot specify PART with OBJECTSET. NGT Recover uses the group definition for the object.
- You cannot use INDEP INTABLESPACE with OBJECTSET.
- You cannot use TABLESPACE with OBJECTSET.

**EXCLUDE (dbname.ts or qualifier.ix)**

Use the EXCLUDE option after an OBJECTSET specification to exclude one or more objects from the rebuild. You can use wildcards or specific names to specify the exclusions. The following wildcards are valid:

- % and * match any number of characters in the object name.
- _ and ? match a single character in the object name.

List the excluded objects following the EXCLUDE keyword. Each item in the list must be in the form `dbname.ts or qualifier.ix` and you must separate the individual items with commas. Enclose the list in parentheses.

**DSNUM**

DSNUM specifies a single data set in the specified space or the entire space for rebuild.

**DSNUM ALL**

DSNUM ALL is the default and specifies that all of the data sets in the space are to be rebuilt. If the space is nonpartitioned and DSNUM ALL is specified or implied, image copies made for specific data sets will only be considered for rebuild if they are registered at the same log point.

**DSNUM integer**

Use this option to rebuild only one partition or data set in the space. For partitioned spaces, DSNUM integer specifies the number (from 1 through 4096) of a single partition in the specified space. For
nonpartitioned spaces, DSNUM integer specifies the number (from 1 to 32) of a single data set of the specified space.

**DSNUM begin : end**

Use this option to rebuild a range of partitions in the space. For partitioned spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

### SCOPE specification

Use the SCOPE options to enable NGT Recover to bypass objects based on specified criteria, thereby defining the scope of the rebuild processing.

To use SCOPE to reduce the scope of an object list, you must first supply the object list. Therefore, REBUILD INDEX SCOPE STATUS (RECP) is not supported.

The SCOPE option is not allowed when NGT Recover is processing clones or RICHK YES is in use.

For the SCOPE specification syntax diagram, see “REBUILD INDEX syntax” on page 216.

#### SCOPE ALL

If you specify SCOPE ALL, NGT Recover rebuilds all of the specified spaces.

#### SCOPE STATUS(status1,status2,...)

NGT Recover bypasses spaces that are not in at least one of the specified statuses.

**Example**

```
REBUILD INDEX OBJECTSET oscreator.osname
   SCOPE STATUS (GRECP,LPL)
```

#### SCOPE PENDING

NGT Recover bypasses spaces that are not in RBDP, RBDP*, PSRBD, or RECP status.

To process spaces with other statuses, use SCOPE STATUS(...).

**Example**

```
REBUILD INDEX OBJECTSET oscreator.osname
   SCOPE PENDING
```
TABLESPACE `databaseName.tableSpaceName`

The TABLESPACE option specifies the table space from which all of the specified indexes are to be rebuilt.

If you do not specify the TABLESPACE option, NGT Recover determines the table space for the first valid index specified. All other specified indexes must belong to the same table space. If you specify INDEX(ALL), you must specify TABLESPACE.

- `databaseName` is the name of the database to which the table space belongs. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE. The default is DSNDB04.

- `tableSpaceName` is the name of the table space from which the indexes are to be rebuilt. BMCUTIL and BMCSYNC, tables that are used by BMC utilities, cannot be used.

AUX

The AUX option specifies if auxiliary objects will be included with the recovery of the base table spaces. For more information, see “AUX” on page 117.

INDEP OUTSPACE

Use the INDEPENDENT (or INDEP) OUTSPACE option to redirect the output of the rebuilding of an index to data sets other than the data sets that are used by DB2. Whether the indexes being rebuilt are specified by index name or by INDEX(ALL), the output for each index is redirected to a different data set.

**Note**

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

You can preallocate the INDEP OUTSPACE data sets using the following guidelines:

- For VCAT-defined spaces, ensure that the volumes or space allocated is large enough to hold the data set.

- If the source space is defined using a STOGROUP, and if primary and secondary quantities are not defined for the source space, NGT Recover uses a sliding-scale calculation for secondary extents, similar to the method DB2 uses.

For more information, see “Primary space allocation” on page 81 and “Secondary extents” on page 81.
You can allow NGT Recover to dynamically allocate the INDEP OUTSPACE data sets for STOGROUP-defined spaces. If the INDEP OUTSPACE data set does not exist, NGT Recover uses the attributes of the source space to dynamically allocate the target INDEP OUTSPACE data set. If the INDEP OUTSPACE data set exists, NGT Recover uses it as is (no DELETE or DEFINE executed).

NGT Recover includes support for multi-data-set spaces. For example, NGT Recover will dynamically allocate A002, A003, to A00n INDEP OUTSPACE data sets as needed.

To have NGT Recover dynamically allocate the INDEP OUTSPACE data set on a specific volume or volumes, pass a volume list by using the STOGROUP...USEORDER specification in the OPTIONS statement (see “STOGROUP specification description” on page 112).

You can specify the name of each independent data set or you can accept the NGT Recover default.

The default name is identical to the DB2 name of the index space to which the indexes are normally directed, but with BMCDBC substituted for DSNDBC:

\[ vcat.BMCDBC.databaseName.indexSpaceName.p0001.znnn \]

The variables are defined as follows:

- **vcat** is the ICF catalog name.
- **databaseName.indexSpaceName** is the name of the database and index space containing the data set being rebuilt.
- **p** corresponds to the value that is used by DB2 for this node.
- **znnn** is the data set number.
  - **z** represents the letter A, B, C, D, or E. These letters correspond to the first digit—0, 1, 2, 3, or 4—of the partition number. **nnn** represents the rest of the partition number. **znnn** represents the partitions numbers as follows:
    - A001 through A999 for partitions 1 through 999
    - B000 through B999 for partitions 1000 through 1999
    - C000 through C999 for partitions 2000 through 2999
    - D000 through D999 for partitions 3000 through 3999
    - E000 through E096 for partitions 4000 through 4096

When you want to specify a name of your own choosing, use the MODEL option to specify the name of the CLUSTER portion of the VSAM data set that you want to use. The data set name that you specify may include any of the symbolic variables that are listed in the description of the MODEL keyword in the following section.
When you use INDEP OUTSPACE, you must specify it prior to the TABLESPACE specification in the REBUILD INDEX command statement, as shown in “REBUILD INDEX syntax” on page 216.

If the rebuilding of a partition is performed on a nonpartitioned index, the entire index must be copied to the data set.

If you are rebuilding an index separately from any recovery of its related table space and you specify INDEP INTABLESPACE, you must also specify INDEP OUTSPACE. NGT Recover will not build DB2 index spaces directly from non-DB2 table data sets.

**Note**
If you do not specify INDEPENDENT INTABLESPACE when you specify the INDEPENDENT OUTSPACE option with REBUILD INDEX, the table space is stopped and started as READ ONLY (RO).

For information about using redirection with multiple NGT Recover commands, see “Example 9: Recovering to a non-DB2 data set” on page 404. For more information about this feature, see “Testing recovery with simulation” on page 52.

**MODEL**

The MODEL option precedes the non-DB2 data set cluster name to which you want to redirect the output from the rebuilding of the specified index. The data set name that you specify may optionally include any of the symbolic variables in “Stacking copies on tape” on page 498.

**INDEP INTABLESPACE**

Use the INDEPENDENT (or INDEP) INTABLESPACE option to specify that the indexes to be rebuilt are to be extracted from a data set other than the table space data set that is used by DB2.

If the independent data set is one of your own choosing that you previously created, you must specify the name of that data set with the MODEL option. If you do not specify a name, NGT Recover assumes the NGT Recover data set name model to construct the data set name:

\[(vcatBMCDBC.databaseName.tableSpaceName.p0001.znnn)\]

The variables are defined as follows:

- **vcat** is the ICF catalog name.
- **databaseName.tableSpaceName** is the name of the database and space containing the data set being rebuilt.
- \( p \) corresponds to the value that is used by DB2 for this node.

- \( znnn \) is the data set number.

\( z \) represents the letter A, B, C, D, or E. These letters correspond to the first digit—0, 1, 2, 3, or 4—of the partition number. \( nnn \) represents the rest of the partition number. \( znnn \) represents the partitions numbers as follows:

- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E096 for partitions 4000 through 4096

If you specify a RECOVER TABLESPACE command (for the table space associated with this index) by using INDEP OUTSPACE in the same SYSIN stream as this REBUILD INDEX command and you also specify INDEP INTABLESPACE, any data set name used must be the same as the name used with INDEP OUTSPACE in the RECOVER TABLESPACE command. If you do not include a data set name, NGT Recover assumes the data set that is named in INDEP OUTSPACE. If you also do not specify INDEP INTABLESPACE, NGT Recover still uses the independent table space data set that is named in the RECOVER TABLESPACE command to extract the keys.

When you use INDEP INTABLESPACE, you must specify it immediately after the TABLESPACE specification, as shown in “REBUILD INDEX syntax” on page 216.

For information about using redirection with multiple NGT Recover commands, see “Example 9: Recovering to a non-DB2 data set” on page 404.

**MODEL**

Use the MODEL option following INDEP INTABLESPACE to specify an existing non-DB2 data set from which the indexes are to be rebuilt. This data set was previously used to receive the output from the recovery of the associated table space.

**PART**

The PART option specifies the partitions of an index on a partitioned table space to rebuild. You can specify the index partition individually in the main body of the REBUILD INDEX command or in a list by using the INDEX (indexName1, indexName2...) option, but you cannot specify the partition in both places. If you specify the partition by using the REBUILD INDEX command statement, you can place the partition specification before or after the TABLESPACE option.
PART ALL

PART ALL is the default and tells NGT Recover to rebuild all partitions of the index space.

PART integer

PART integer tells NGT Recover to rebuild the index space for the specified partition. If you specify a nonpartitioned index, the index entries are replaced for the named partition.

integer must be in the range 1 through 4096.

PART begin : end

Use this option to recover a range of partitions for the index space. For partitioned index spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

CLONE

The CLONE option indicates that NGT Recover is to rebuild clone index data. When you work with clone data, the following limitations apply:

■ You cannot refer to the same object with the CLONE option and without the CLONE option in the same recovery step; you cannot process the base and its clone in the same command.

■ Related objects (table space and all indexes for its table, all related LOB objects) should use the same CLONE specification in the same recovery step.

WORKDDN DDName

The WORKDDN DDName option specifies the work data set ddname.

The default is SYSUT1 but may change to UNLOADnn for table spaces with fewer than 100 partitions when REBUILD INDEX is accompanied by one or more RECOVER UNLOADKEYS statements. For table spaces with 100 or more partitions, the default value is UNLOAnnn. WORKDDN is required when REBUILD INDEX is accompanied by RECOVERUNLOADKEYS for the same partition and NOUKSORT is specified.

When you use multiple REBUILD INDEX command statements, the following restrictions apply to the use of WORKDDN:
If you use multiple command statements for a *nonpartitioned* table space, the *DDName* that you specify for WORKDDN must be the same in all of the statements.

If you use multiple command statements for a *partitioned* table space and request the rebuilding of a nonpartitioned index, the *DDName* that you specify for WORKDDN must be the same in all of the statements. Otherwise, NGT Recover cannot extract keys for all indexes from each partition in one pass of the table space.

If you specify only the rebuilding of a partitioned index for a partitioned table space with multiple REBUILD INDEX command statements, the value specified for WORKDDN has no restrictions.

WORKDDN can not be specified for a partitioned index if you use RECOVER UNLOADKEYS for the space.

**Note**

NGT Recover ignores WORKDDN if the value of the MAXKSORT option is greater than 1. The value of MAXKSORT is set in the installation options or on the OPTION command. For more information about MAXKSORT, see “MAXKSORT integer” on page 134 and “MAXKSORT” on page 587.

If you specify RECOVERUNLOADKEYS and NOUKSORT together with REBUILD INDEX for the same table space and partitions, the WORKDDN ddname on both statements must be the same. The ddname is considered a prefix and must not be more than six characters for spaces with fewer than 100 partitions or not more than five characters for spaces with 100 or more partitions.

For more information, see “Restrictions for key work data sets, SKEYDDN, SORTNUM, and SORTDEVT” on page 91.

**NOWORKDDN**

The NOWORKDDN option causes the extracted keys to be sent directly to the sort routine without first writing these keys to a work data set. Using this option causes the index to rebuild faster. However, restartability is more limited because, without the work data set, restarting at the beginning of the BUILD phase is not possible. The restart will have to redo the UNLOAD phase.

The keys are sent directly to the sort if neither WORKDDN nor NOWORKDDN is specified, and no SYSUT1 DD statement exists in the JCL.

If you use RECOVER UNLOADKEYS for the space, NOWORKDDN is the default for a partitioned index.
SORTDEVT and SORTNUM sort work data set options

To sort index keys, NGT Recover invokes BMCSORT, which allocates the required temporary sort work data sets. You can take one of the following courses of action:

- Specify SORTNUM and SORTDEVT to direct the allocation.
- Specify neither SORTNUM nor SORTDEVT, and let BMCSORT allocate the work data sets according to sort rules.
- Specify neither SORTNUM nor SORTDEVT, and specify SxxxWKnn DD statements in the JCL.
- Specify only SORTDEVT, and let BMCSORT determine the number of data sets.

If you use SxxxWKnn DD statements in the JCL, any SORTDEVT and SORTNUM specifications that are present are ignored. If you do not use SxxxWKnn DD statements and do not specify SORTDEVT or SORTNUM, the sort routine uses the installation options for SORTDEVT and SORTNUM.

When you use multiple REBUILD INDEX command statements, the following restrictions apply to the use of SORTNUM and SORTDEVT:

- If you use multiple command statements for a nonpartitioned table space, the values that you use for SORTNUM and SORTDEVT must be the same in all of the statements because NGT Recover unloads the keys concurrently for all of the indexes.
- If you use multiple command statements for a partitioned table space and you specify a nonpartitioned index recovery, the values that you use for SORTNUM and SORTDEVT must be the same in all of the statements so NGT Recover can extract keys for all indexes from each partition in one pass of the table space.

If you specify the rebuilding of a partitioned index for a partitioned table space with multiple REBUILD INDEX command statements, the values specified for SORTNUM and SORTDEVT have no restrictions.

SORTDEVT deviceType

SORTDEVT deviceType specifies the device type for the temporary sort work data sets that BMCSORT uses.

If you specify SORTDYN NO ("SORTDYN" on page 114), BMCSORT defaults are used even if you specify a value for SORTDEVT. Otherwise, NGT Recover uses its internal default value, SYSDA, if you do not specify SORTDEVT.
SORTNUM integer

The SORTNUM option affects the allocation of sort work files when BMCSORT is allocating your sort work files dynamically. You can specify an integer value of 1 through 255.

When you specify this option, BMCSORT dynamically allocates the number of sort work files that it needs for each sort task up to the maximum that is illustrated in the following formula:

\[
\text{maximum dynamically allocated sort work files} = n - \text{preallocated sort work files}
\]

If you specify integer from 1 through 32, \( n \) equals 32. If you specify integer greater than 32, \( n \) equals integer.

**Note**
Preallocated sort work files include sort work files that are allocated in your JCL.

If you do not specify a value for SORTNUM, NGT Recover uses the installation option default.

DEFINE

DEFINE specifies whether DEFINE NO objects will be instantiated.

**Note**
IBM DB2 Version 11 or later supports the DEFINE option.

DEFINE YES

NGT Recover instantiates DEFINE NO objects, if necessary, and updates the catalog and DBD to reflect that the object has been defined.

DEFINE NO

NGT Recover does not process DEFINE NO objects. NO is the default value.

NUMREC

NUMREC specifies the size of the sort file for the index sort routine. This option is appropriate only when NOWORKDDN is specified because the utility counts records when a work data set is used to collect the keys before the sort begins. The size is defined as an estimate or the exact number of key records to process.
For information about performance implications, see “NGT Recover concepts” on page 451.

**Tip**

NGT Recover usually determines a good estimate for the number of key records automatically. BMC recommends that in most cases you do not specify NUMREC or that you specify NUMREC CALC (the default).

**NUMREC CALC**

NUMREC CALC is the default value.

Specifying this option causes NGT Recover to estimate the number of key records to be sorted. NGT Recover uses RUNSTATS statistics, if available, to make these estimates. If these statistics are not available, NGT Recover tries to estimate the number of key records based on the number of pages in the associated table space.

NGT Recover uses a minimum estimate of 400,000 for each index key sort (or 200,000 if the key length is greater than 1000 characters).

**NUMREC NOEST**

When you specify NUMREC NOEST, NGT Recover passes an estimate of 400,000 to each index key sort (or 200,000 if the key length is greater than 1000 characters).

**NUMREC EST integer**

Use NUMREC EST integer to specify an estimated number of key records for the index sort routine. integer must be a positive integer.

If you have specified multiple indexes on the REBUILD or RECOVER UNLOADKEYS statement, NGT Recover divides the NUMREC value equally among the indexes. In general, NUMREC EST is unnecessary and might degrade performance unless the specified value is accurate.

If the specified estimate is less than 400,000 (200,000 for keys with a length greater than 1000 characters), NGT Recover uses an estimate of 400,000 (200,000 if the key length is greater than 1000 characters).

**NUMREC ABS integer**

Use NUMREC ABS integer to specify the exact number of key records for the index sort routine. integer must be a positive integer.
If integer is a value that does not represent the true number of key records passed to the index sort routine, the index sort routine abends. Use extreme caution when you specify NUMREC ABS.

### ANALYZE

The ANALYZE option prints the index rebuilding plan before executing that plan.

Information about the following items used for rebuilding the index is included in the plan:

- The record sizes for index sort work data sets
- The phases to occur during execution
- The steps to occur within each phase

Use this information to allocate space more accurately for index sort work data sets and index key work data sets, thereby limiting abends that are caused by inadequate data set allocations. You can use these statistics with historical information about the times required to perform the various operations to estimate the time required to rebuild the indexes.

**Note**

If you coded ANALYZE with the OPTIONS command, you must code the same value here. If you use multiple REBUILD INDEX command statements you must use the same value for the ANALYZE option in all of the statements. If you specify a value for ANALYZE on one command statement and take the default on the others, the value that you specify on the single command statement becomes the default for the others.

### ANALYZE YES

ANALYZE YES is the default and provides all of the information described in the preceding section.

### ANALYZE NO

If you specify ANALYZE NO, NGT Recover provides no information.

### ANALYZE ONLY

If you specify ANALYZE ONLY, NGT Recover provides the same information as it provides with ANALYZE YES and then stops the job when the ANALYZE phase is complete. You can use the data from the ANALYZE phase to determine the
resources required and what will happen during the recovery job. You cannot restart a NGT Recover job that specifies ANALYZE ONLY, but you can start a new recovery job.

Use the ANALYZE ONLY option to determine which phases will occur with a specific request, or which copy data sets, log data sets, and change accumulation data sets are used in the rebuilding of the indexes.

**REDEFINE**

Use the REDEFINE option to avoid the normal deletion and reallocation of the index space data sets for STOGROUP-defined index spaces. This option is ignored for VCAT-defined index space data sets.

The value that you specify for the REDEFINE option applies to all STOGROUP-defined index spaces included in a single REBUILD INDEX command statement. To rebuild multiple STOGROUP-defined index spaces with some reallocated and others not reallocated, you must use one REBUILD INDEX command statement for the index spaces to be reallocated and a second REBUILD INDEX command statement for those index spaces not to be reallocated.

Regardless of the value of REDEFINE, if the data sets for a STOGROUP-defined index space do not exist, NGT Recover creates the data sets.

---

**Note**

When you are rebuilding a multi-data-set, nonpartitioned index space, more or fewer data sets may result than were previously used.

---

**REDEFINE YES**

REDEFINE YES is the default. If you specify REDEFINE YES, NGT Recover deletes and reallocates the VSAM data sets for STOGROUP-defined index spaces prior to the start of the BUILD phase. For more information, see Storage group-defined data sets on page 80.

**NOSCRATCH**

If you specify NOSCRATCH with REDEFINE YES, NGT Recover issues IDCAMS to uncatalog without scratching. You can use NOSCRATCH to support disaster recovery scenarios where the following items are true:

- DASD volumes at the recovery site differ from those at the primary site.

- The operating system catalog and the DB2 catalog and directory are restored before running NGT Recover.

- DB2 STOGROUPS are changed to point to the recovery site packs.
Using this option eliminates mount messages for the primary site packs that do not exist at the recovery site.

**Note**
REDEFINE YES has no effect if you specify INDEP OUTSPACE. For the instructions for creating data sets for INDEP OUTSPACE, see “INDEP OUTSPACE” on page 223.

**REDEFINE NO**

REDEFINE NO tells NGT Recover not to delete and reallocate the VSAM data sets for the index space. Use this option to treat STOGROUP-defined index spaces as VCAT-defined index spaces and to reallocate the VSAM data sets manually. New volumes are used as needed from the STOGROUP, even with REDEFINE NO specified.

**REUSE**

The REUSE option specifies that the space will not be deleted or redefined. This option cannot be used with the REDEFINE option.

**NOCOPYPEND**

If you specify the NOCOPYPEND option, NGT Recover resets ICOPY status on COPY YES indexes and issues message BMC96232 to inform you that ICOPY status has been reset even though the space is not recoverable.

**SHRLEVEL**

The SHRLEVEL option specifies the level of access that concurrently operating DB2 applications and utilities should have to the target table space or index space. You can specify read-only access or read-write access. If you do not specify this option, NGT Recover allows read-only access to the target table space during the rebuild process.

**SHRLEVEL REFERENCE**

SHRLEVEL REFERENCE is the default and allows read-only access by other programs to the target space during the rebuild process.

**SHRLEVEL CHANGE**

SHRLEVEL CHANGE allows read-write application access to DB2 table space and index space objects during the rebuild process.
Performing an online rebuild index by specifying SHRLEVEL CHANGE offers the following benefits:

- Allows full access to DB2 data during most of the rebuild
- Delivers significantly improved data availability by greatly reducing the outage for the DB2 objects
- Operates in a nondestructive manner, which allows you to make the objects available without having to recover in the event of a failure

**Note**

Online index rebuild is a feature of the BMC Recovery Management for DB2 solution. Using the SHRLEVEL CHANGE option to perform an online index rebuild requires a valid Recovery Management solution password.

---

**MAXRO**

MAXRO specifies the maximum length of time for the last iteration of log processing.

*integer*

This integer specifies the maximum number of seconds that NGT Recover spends applying log records for the last iteration of log processing. Specify this value as a positive integer (0 or greater). During that iteration, applications have no access. The accrual execution time of the last iteration might exceed the specified value for MAXRO.

Specifying a small positive value minimizes the period of no access, but it might increase the elapsed time for REBUILD INDEX to complete. If you specify a very large positive value, the second iteration of log processing will probably be the last iteration.

The default value is the value of the lock timeout system parameter IRLMRWT.

**DEFER**

This value tells NGT Recover to continue applying log records indefinitely, regardless of the value set in any other SHRLEVEL CHANGE option.

**LONGLOG**

LONGLOG specifies the action to take if NGT Recover determines that a longlog condition exists. A longlog condition exists when NGT Recover determines over a period of time that the DB2 subsystem is generating log records for the objects that you are rebuilding faster than NGT Recover is
applying the records. After detecting that a longlog condition exists, NGT Recover sends an action write-to-operator (WTO) notification and continues to apply the log records until the delay interval specified on the DELAY option expires. If the longlog condition still exists, NGT Recover takes the action that you specify with one of the following options.

CONTINUE

CONTINUE indicates that NGT Recover is to continue processing.

TERM

TERM indicates that NGT Recover is to terminate the reorganization as specified by DELAY.

DRAIN

DRAIN indicates that NGT Recover is to drain the write claim class specified by DELAY, which applies the remaining log records.

DELAY

DELAY specifies the number of seconds that are to elapse from the time NGT Recover detects a longlog condition until it performs the action specified by the LONGLOG option. If the longlog condition no longer exists at the end of the time period specified by DELAY, the timer is reset. When NGT Recover detects the next longlog condition, it restarts the timer, using the original DELAY value. The value must be a positive integer (0 or greater). The default value is 1200.

DRAIN_WAIT

DRAIN_WAIT specifies the number of seconds that REBUILD INDEX is to wait when draining the table space or index. The specified time is the aggregate time for objects that are to be checked.

integer can be any integer from 0 to 1800. If you do not specify DRAIN_WAIT or if you specify a value of 0, NGT Recover uses the value of the lock timeout subsystem parameter IRLMRWT.

RETRY

RETRY specifies the maximum number of retries that REBUILD INDEX should attempt.

integer can be any integer from 0 to 255. REBUILD INDEX uses 1 as the default if you do not specify a value.

RETRY_DELAY

RETRY_DELAY specifies the minimum number of seconds between retries.
integer can be any integer from 1 to 1800.

If you do not specify RETRY_DELAY, REBUILD INDEX uses the DRAIN_WAIT value x RETRY value.

**RBALRSN_CONVERSION**

The RBALRSN_CONVERSION option specifies the RBA or LRSN format of the index.

If this option is not specified, UTILITY_OBJECT_CONVERSION from the DSNZPARM determines the RBA or LRSN format. RBALRSN_CONVERSION cannot be used with indexes on table spaces that are involved in a clone relationship.

**NONE**

NONE tells NGT Recover to rebuild the index with the current format defined by the column RBA_FORMAT in SYSIBM.SYSINDEXES.

**BASIC**

BASIC tells NGT Recover to convert the RBA or LRSN format to the 6-byte format if the original format is EXTENDED. The job will fail if UTILITY_OBJECT_CONVERSION is set to NOBASIC.

**EXTENDED**

EXTENDED tells NGT Recover to convert the RBA or LRSN format to the 10-byte format if the original format is BASIC.

**RECOVER UNLOADKEYS**

This topic describes the syntax of the RECOVER UNLOADKEYS command and its options.

---

*Note*

RECOVER UNLOADKEYS ignores the MAXKSORT option and processes as if you specified MAXKSORT=1. If you also specify a REBUILD INDEX for the partitioning index and specify a MAXKSORT value of at least 3, NGT Recover uses a separate sort for the partitioning index and does the REBUILD INDEX for the partitioning index in parallel with RECOVER UNLOADKEYS.
RECOVER UNLOADKEYS syntax

The following figure shows the syntax of the RECOVER UNLOADKEYS command.

Figure 35: RECOVER UNLOADKEYS syntax diagram

RECOVER UNLOADKEYS option descriptions

NGT Recover provides the following options for use with the RECOVER UNLOADKEYS command. The options are described in the order in which they are shown in the RECOVER UNLOADKEYS syntax diagram.

1 You cannot specify PART in both places.
RECOVER

Use the RECOVER command when you want to perform an actual recovery.

SIMRCVR

SIMRCVR is a synonym for the RECOVER command. Use the SIMRCVR keyword to clarify that you are running in simulation mode. When you use SIMRCVR, you must also specify OPTIONS SIMULATE YES, which requires a valid BMC Recovery Management for DB2 solution password.

For more information, see “SIMULATE YES” on page 132.

UNLOADKEYS

UNLOADKEYS specifies one or more indexes for which keys and row IDs are to be extracted.

All specified indexes must be nonpartitioned indexes on the table of a partitioned table space.

(indexName1, indexName2, . . . . . )

This specifies each of the indexes for which keys and row IDs are to be extracted. Each index name is in the form authid.indexName. If you do not provide the authid qualifier, the qualifier is defaulted to the user ID of the user executing the utility. You must enclose the list of qualified names in parentheses.

Note

RECOVER UNLOADKEYS does not support indexes defined using an expression.

PART integer

This parameter specifies the partition of the index for which keys and row IDs are to be extracted.

PART begin : end

Use this option to recover a range of partitions for the index space. For partitioned index spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.
**ALL**

UNLOADKEYS(ALL) specifies that keys and row IDs are to be extracted for all nonpartitioned indexes on the table space named in the TABLESPACE option.

**TABLESPACE databaseName.tableSpaceName**

The TABLESPACE option specifies the partitioned table space from which keys and row IDs are to be extracted for the nonpartitioned indexes specified by the UNLOADKEYS option. If you specify UNLOADKEYS(ALL), you must also specify TABLESPACE. If you specify UNLOADKEYS(indexName1, . . . . . ) and do not specify TABLESPACE, NGT Recover determines the table space for the first index specified. All other specified indexes must belong to the same table space.

- `databaseName` is the name of the database to which the table space belongs. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE

- `tableSpaceName` is the name of the table space from which nonpartitioned index keys are to be extracted. BMCUTIL and BMCSYNC, tables that are used by BMC utilities, cannot be used.

**PART**

The PART option specifies the partitions of an index on a partitioned table space to unload. You can specify the index partition individually in the main body of the RECOVER UNLOADKEYS command or in a list by using the INDEX (indexName1, indexName2...) option, but you cannot specify the partition in both places. If you specify the partition by using the RECOVER UNLOADKEYS command statement, you can place the partition specification before or after the TABLESPACE option.

**PART integer**

`PART integer` specifies the partition of the table space from which the keys and row IDs are extracted. You must always specify this option with RECOVER UNLOADKEYS.

`integer` must be in the range 1 through 4096.

You can place the partition specification before or after the TABLESPACE option. You can specify a partition individually in the main body of the RECOVER UNLOADKEYS command statement or in a list by using the UNLOADKEYS (indexName1, indexName2...) option, but you cannot specify the partition in both places.
PART begin : end

Use this option to recover a range of partitions for the index space. For partitioned index spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

INDEPENDENT INTABLESPACE

Use the INDEPENDENT (or INDEP) INTABLESPACE option to specify that the nonpartitioned index keys and row IDs are to be extracted from a table space data set other than the one that is used by DB2. If the independent data set is one of your own choosing that you previously created, you must specify the name of that data set with the MODEL option. If you do not specify a name, NGT Recover assumes the NGT Recover data set name model to construct the data set name:

\[(vcat.BMCDBC.databaseName.indexSpaceName.p0001.znnn)\]

The variables are defined as follows:

- \(vcat\) is the ICF catalog name.
- \(databaseName.indexSpaceName\) is the name of the database and index space containing the data set being rebuilt.
- \(p\) corresponds to the value that is used by DB2 for this node.
- \(znnn\) is the data set number.

\(z\) represents the letter A, B, C, D, or E. These letters correspond to the first digit—0, 1, 2, 3, or 4—of the partition number. \(nnn\) represents the rest of the partition number. \(znnn\) represents the partitions numbers as follows:

- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E096 for partitions 4000 through 4096

If you specify a RECOVER TABLESPACE command (for the table space associated with this index) by using INDEP OUTSPACE in the same SYSIN stream as this RECOVER UNLOADKEYS command and you specify INDEP INTABLESPACE, any data set name specified must be the same as the name used
with INDEP OUTSPACE in the RECOVER TABLESPACE command. If you do not include a data set name, NGT Recover assumes that data set that is named in INDEP OUTSPACE. If you also do not specify INDEP INTABLESPACE, NGT Recover still uses the independent table space data set that is named in the RECOVER TABLESPACE command to extract the keys.

When you use INDEP INTABLESPACE, you must specify it immediately after the TABLESPACE specification, as shown in “RECOVER UNLOADKEYS syntax” on page 238.

For information about using redirection with multiple NGT Recover commands, see “Example 9: Recovering to a non-DB2 data set” on page 404. For more information about some applications of this feature, see “Testing recovery with simulation” on page 52.

**MODEL**

Use the MODEL option following INDEP INTABLESPACE to specify an existing non-DB2 data set from which the nonpartitioned index keys and row IDs are to be extracted. This non-DB2 data set was previously used to receive the output from the recovery of the associated table space.

**CLONE**

The CLONE option indicates that NGT Recover is to recover clone table or index data. When you recover with clone data, the following limitations apply:

- You cannot refer to the same object with the CLONE option and without the CLONE option in the same recovery step; you cannot process the base and its clone in the same command.

- Related objects (table space and all indexes for its table, all related LOB objects) should use the same CLONE specification in the same recovery step.

**SKEYDDN**

Use the SKEYDDN option to specify the ddname for the unloaded keys work data set that is to be used to hold the sorted keys and row IDs. The default ddname is SKEY. The maximum length of the ddname specified by SKEYDDN is eight characters. Certain restrictions apply to this option, as follows:

- If you use multiple RECOVER UNLOADKEYS command statements for a table space, the ddname that you specify for SKEYDDN must be the same in all of the statements.

- If you use other RECOVER UNLOADKEYS command statements for other table spaces or other REBUILD INDEX command statements for other table spaces, any value you specify for SKEYDDN with those statements must be different.
If you use RECOVER UNLOADKEYS and REBUILD INDEX command statements for a table space, REBUILD INDEX defaults to NOWORKDDN if you specify SKEYDDN.

Do not use data sets created with SIMRCVR UNLOADKEYS except with SIMRCVR BUILDINDEX.

**SORTDEVT and SORTNUM sort work data set options**

To sort index keys, NGT Recover invokes BMCSORT, which allocates the required temporary sort work data sets. You can take one of the following courses of action:

- Specify SORTNUM and SORTDEV to direct the allocation.
- Specify neither SORTNUM nor SORTDEV, and let BMCSORT allocate the work data sets according to sort rules.
- Specify neither SORTNUM nor SORTDEV, and specify SxxxWKnn DD statements in the JCL.
- Specify only SORTDEV, and let BMCSORT determine the number of data sets.

If you use SxxxWKnn (sort work) DD statements in the JCL, any SORTDEV and SORTNUM specifications that are present are ignored. If you do not use SxxxWKnn DD statements and do not specify SORTDEV or SORTNUM, the sort routine uses the installation options for SORTDEV or SORTNUM.

**SORTDEVT deviceType**

SORTDEVT deviceType specifies the device type for the temporary sort work data sets that BMCSORT uses.

If you specify SORTDYN NO ("SORTDYN" on page 114), BMCSORT defaults are used even if you specify a value for SORTDEV. Otherwise, NGT Recover uses its internal default value, SYSDA, if you do not specify SORTDEV.

**SORTNUM integer**

The SORTNUM option affects the allocation of sort work files when BMCSORT is allocating your sort work files dynamically. You can specify an integer value of 1 through 255.

When you specify this option, BMCSORT dynamically allocates the number of sort work files that it needs for each sort task up to the maximum that is illustrated in the following formula:

\[
\text{maximum dynamically allocated sort work files} = n - \text{preallocated sort work files}
\]
If you specify integer from 1 through 32, \( n \) equals 32. If you specify integer greater than 32, \( n \) equals integer.

**Note**
Preallocated sort work files include sort work files that are allocated in your JCL.

If you do not specify a value for SORTNUM, NGT Recover uses the installation option default.

**NUMREC**

You must specify the same NUMREC values for all RECOVER UNLOADKEYS command statements for the same table space. NUMREC specifies the size of the sort file for the index sort routine. The size is defined as an estimate or the exact number of key records to process.

For information about performance implications, see “NGT Recover concepts” on page 451.

**Tip**
NGT Recover usually determines a good estimate for the number of key records automatically. BMC recommends that in most cases you do not specify NUMREC or that you specify NUMREC CALC (the default).

**NUMREC CALC**

NUMREC CALC is the default value.

Specifying this option allows NGT Recover to estimate the number of key records to be sorted. NGT Recover uses RUNSTATS statistics, if available, to make these estimates. If these statistics are not available, NGT Recover tries to estimate the number of key records based on the number of pages in the associated table space.

NGT Recover uses a minimum estimate of 400,000 for each index key sort (or 200,000 if the key length is greater than 1000 characters).

**NUMREC NOEST**

When you specify NUMREC NOEST, NGT Recover passes an estimate of 400,000 to each index key sort (or 200,000 if the key length is greater than 1000 characters).

**NUMREC EST integer**

Use NUMREC EST integer to specify an estimated number of key records for the index sort routine. integer must be a positive integer.
If you have specified multiple indexes on the REBUILD or RECOVER UNLOADKEYS statement, NGT Recover divides the NUMREC value equally among the indexes. In general, NUMREC EST is unnecessary and might degrade performance unless the specified value is accurate.

If the specified estimate is less than 400,000 (200,000 for keys with a length greater than 1000 characters), NGT Recover uses an estimate of 400,000 (200,000 if the key length is greater than 1000 characters).

**NUMREC ABS integer**

Use NUMREC ABS integer to specify the exact number of key records for the index sort routine. integer must be a positive integer.

**WARNING**

If integer is a value that does not represent the true number of key records passed to the index sort routine, the index sort routine will abend. Use extreme caution when you specify NUMREC ABS.

**ANALYZE**

The ANALYZE option prints a recovery plan before executing that plan.

Information about the following items used for recovery is included in the plan:

- The phases to occur during execution
- The steps to occur within each phase

Use this information to allocate space more accurately for the unload work data sets so limiting abends that are caused by inadequate data set allocations. You can use these statistics with historical information about the times required to perform the various operations to estimate recovery time.

**Note**

If you coded ANALYZE with the OPTIONS command, you must code the same value here. If you use multiple NGT Recover command statements, you must use the same value for the ANALYZE option in all of the statements. If you specify a value for ANALYZE on one command statement and take the default on the others, the value you specify on the single command statement becomes the default for the others.

**ANALYZE YES**

ANALYZE YES is the default and provides information about the items used for recovery that are in the recovery plan.
ANALYZE NO

If you specify ANALYZE NO, NGT Recover provides no information.

ANALYZE ONLY

If you specify ANALYZE ONLY, NGT Recover provides the same information as it provides with ANALYZE YES and then stops the job when the ANALYZE phase is complete. You can use the data from the ANALYZE phase to determine the resources required and what will happen during the recovery job. You cannot restart a NGT Recover job that specifies ANALYZE ONLY, but you can start a new recovery job.

Use the ANALYZE ONLY option to determine which phases will occur with a specific recovery request.

RECOVER BUILDINDEX

This topic describes the syntax of the RECOVER BUILDINDEX command and its options.

Note

RECOVER BUILDINDEX ignores the MAXKSORT option and processes as if you specified MAXKSORT=1.
RECOVER BUILDINDEX syntax

The following figure shows the syntax of the RECOVER BUILDINDEX command.

**Figure 36: RECOVER BUILDINDEX syntax diagram**

RECOVER BUILDINDEX option descriptions

NGT Recover provides the following options for use with the RECOVER BUILDINDEX command. The options are described in the order in which they are shown in the RECOVER BUILDINDEX syntax diagram.

**RECOVER**

Use the RECOVER command when you want to perform an actual recovery.

**SIMRCVR**

SIMRCVR is a synonym for the RECOVER command. Use the SIMRCVR keyword to clarify that you are running in simulation mode. When you use SIMRCVR, you must also specify OPTIONS SIMULATE YES, which requires a valid BMC Recovery Management for DB2 solution password.
For more information, see “SIMULATE YES” on page 132.

**BUILDINDEX**

The BUILDINDEX option specifies one or more nonpartitioned indexes (and so identifies the corresponding index spaces) to be rebuilt. The keys for the indexes must have already been extracted in RECOVER UNLOADKEYS jobs and be available in the work data sets (the unload data sets) specified in that job.

(indexName1,indexName2, . . . . . )

This parameter specifies each of the nonpartitioned indexes that you want to rebuild. Each index name is in the form authid.indexName. If you do not provide the authid qualifier, the qualifier is defaulted to the user ID of the user executing the utility. The keys and row IDs for all of the indexes must be already contained in the unload data sets specified in the job.

You must enclose the list of qualified names in parentheses.

**ALL**

BUILDINDEX(ALL) specifies that all nonpartitioned indexes on the table space are to be rebuilt and expects that the keys and row IDs are contained in the unload data sets specified in the job.

**INDEPENDENT OUTSPACE**

Use the INDEPENDENT (or INDEP) OUTSPACE option to redirect the output of a RECOVER BUILDINDEX run to data sets other than those that are used by DB2.

---

**Note**

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

---

You can preallocate the INDEP OUTSPACE data sets using the following guidelines:

- For VCAT-defined spaces, ensure that the volumes or space allocated is large enough to hold the data set.

- If the source space is defined using a STOGROUP, and if primary and secondary quantities are not defined for the source space, NGT Recover uses a sliding-scale calculation for secondary extents, similar to the method DB2 uses.
For more information, see “Primary space allocation” on page 81 and “Secondary extents” on page 81.

You can allow NGT Recover to dynamically allocate the INDEP OUTSPACE data sets for STOGROUP-defined spaces. If the INDEP OUTSPACE data set does not exist, NGT Recover uses the attributes of the source space to dynamically allocate the target INDEP OUTSPACE data set. If the INDEP OUTSPACE data set exists, NGT Recover uses it as is (no DELETE or DEFINE executed).

NGT Recover includes support for multi-data-set spaces. For example, NGT Recover will dynamically allocate A002, A003, to A00 n INDEP OUTSPACE data sets as needed.

To have NGT Recover dynamically allocate the INDEP OUTSPACE data set on a specific volume or volumes, pass a volume list by using the STOGROUP..USEORDER specification in the OPTIONS statement (see “STOGROUP specification description” on page 112).

You can specify the name of each independent data set or you can accept the NGT Recover default.

The default name is identical to the DB2 name of the index space to which the indexes are normally directed but with BMCDBC substituted for DSNDBC:

\[vcat.BMCDBC.databaseName.indexSpaceName.p0001.znnn\]

The variables are defined as follows:

- \(vcat\) is the ICF catalog name.
- \(databaseName.indexSpaceName\) is the name of the database and index space containing the data set being rebuilt.
- \(p\) corresponds to the value that is used by DB2 for this node.
- \(znnn\) is the data set number.

\(z\) represents the letter A, B, C, D, or E. These letters correspond to the first digit–0, 1, 2, 3, or 4–of the partition number. \(nnn\) represents the rest of the partition number. \(znnn\) represents the partitions numbers as follows:

- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E096 for partitions 4000 through 4096

Any data set name that you specify must be preceded by the MODEL keyword.
When you want to specify a name of your own choosing, use the MODEL option to specify the name of the CLUSTER portion of the VSAM data set that you want to use. The data set name that you specify may include any of the symbolic variables that are listed in the description of the MODEL keyword in the following section.

When you use INDEP OUTSPACE, you must specify it prior to the TABLESPACE specification in the RECOVER BUILDINDEX command statement, as shown in “RECOVER BUILDINDEX syntax” on page 247.

For information about using redirection with multiple NGT Recover commands, see “Example 9: Recovering to a non-DB2 data set” on page 404. For information about some applications of this feature, see “Testing recovery with simulation” on page 52.

**MODEL**

The MODEL option precedes the non-DB2 data set cluster name to which you want to redirect the output from the rebuilding of the specified index. The data set name that you specify may optionally include any of the symbolic variables in “Stacking copies on tape” on page 498.

**TABLESPACE databaseName.tableSpaceName**

The TABLESPACE option specifies the partitioned table space from which the index keys and row IDs contained in the unload data sets were extracted. If you specify BUILDINDEX(ALL), you must also specify TABLESPACE. If you specify BUILDINDEX(indexName1,. . . . . .) and do not specify TABLESPACE, NGT Recover determines the table space for the first index specified. All other specified indexes must belong to that table space.

- *databaseName* is the name of the database to which the table space belongs. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE.

- *tableSpaceName* is the name of the table space for which all nonpartitioned indexes are to be rebuilt. BMCUTIL and BMCSYNC, tables that are used by BMC utilities, cannot be used.

**CLONE**

The CLONE option indicates that NGT Recover is to recover clone table or index data. When you recover with clone data, the following limitations apply:

- You cannot refer to the same object with the CLONE option and without the CLONE option in the same recovery step; you cannot process the base and its clone in the same command.
Related objects (table space and all indexes for its table, all related LOB objects) should use the same CLONE specification in the same recovery step.

**SKEYDDN**

The SKEYDDN option specifies a prefix for the ddnames of the unloaded keys work data sets created by prior RECOVER UNLOADKEYS job steps. The unload work data sets must contain all of the keys needed to rebuild all of the indexes requested. Any ddname within the JCL beginning with the specified prefix is recognized as an unload work data set. The remainder of the ddname is not restricted to numeric values. However, the maximum length of the prefix specified by SKEYDDN is eight characters. The default value is SKEY.

Do not use data sets created with SIMRCVR UNLOADKEYS except with SIMRCVR BUILDINDEX.

**ANALYZE**

The ANALYZE option prints a recovery plan before executing that plan.

Information about the following items used for recovery is included in the plan:

- The record sizes for index sort work data sets
- The phases to occur during execution
- The steps to occur within each phase

Use this information to allocate space more accurately for index sort work data sets and index key work data sets, thereby limiting abends that are caused by inadequate data set allocations. You can use these statistics with historical information about the times required to perform the various operations to estimate recovery time.

**ANALYZE YES**

ANALYZE YES is the default and provides all of the information in the recovery plan.

**ANALYZE NO**

If you specify ANALYZE NO, NGT Recover provides no information.

**ANALYZE ONLY**

If you specify ANALYZE ONLY, NGT Recover provides the same information as it provides with ANALYZE YES and then stops the job when the ANALYZE phase is complete. You can use the data from the ANALYZE phase to determine the
Use the ANALYZE ONLY option to determine which phases will occur with a specific recovery request.

**REDEFINE**

Use the **REDEFINE** option to avoid the normal deletion and reallocation of the index space data sets for STOGROUP-defined index spaces. This option is ignored for VCAT-defined index space data sets.

The value that you specify for **REDEFINED** applies to all STOGROUP-defined index spaces included in a single **RECOVER BUILDINDEX** command statement. To rebuild multiple STOGROUP-defined index spaces with some reallocated and others not reallocated, you must use one **RECOVER BUILDINDEX** command statement for the index spaces to be reallocated and a second **RECOVER BUILDINDEX** command statement (in a separate execution of NGT Recover) for those index spaces not to be reallocated.

Regardless of the value of **REDEFINED**, if the data sets for a STOGROUP-defined index space do not exist, NGT Recover allocates the VSAM data sets for the index space.

**Note**

When a **RECOVER BUILDINDEX** of a multi-data-set index space is performed, more or fewer data sets may result than were previously used.

**REDEFINED YES**

**REDEFINED YES** is the default. If you specify **REDEFINED YES**, NGT Recover deletes and reallocates the VSAM data sets for STOGROUP-defined index spaces prior to the start of the BUILD phase. For more information, see Storage group-defined data sets on page 80.

**NOSCRATCH**

If you specify **NOSCRATCH** with **REDEFINED YES**, NGT Recover issues IDCAMS to uncatalog without scratching. You can use **NOSCRATCH** to support disaster recovery scenarios where the following items are true:

- The DASD volumes at the recovery site differ from those at the primary site.
- The operating system catalog and the DB2 catalog and directory are restored prior to running NGT Recover.
DB2 STOGROUPS are changed to point to the recovery site packs.

Using this option eliminates mount messages for the primary site packs that do not exist at the recovery site.

**Note**

REDEFINE YES has no effect if INDEP OUTSPACE is coded. For instructions for creating data sets for INDEP OUTSPACE, see “INDEPENDENT OUTSPACE” on page 248.

---

**REDEFINE NO**

REDEFINE NO tells NGT Recover not to delete and reallocate the VSAM data sets for the index space. Use this option to treat STOGROUP-defined index spaces as VCAT-defined index spaces and to reallocate the VSAM data sets manually. New volumes are used as needed from the STOGROUP, even with REDEFINE NO specified.

**REUSE**

The REUSE option specifies that the space will not be deleted or redefined. This option cannot be used with the REDEFINE option.

---

**LOGSCAN**

This topic describes the syntax of the LOGSCAN command and its options.
LOGSCAN syntax

The following figures show the syntax of the LOGSCAN command.

Figure 37: LOGSCAN syntax diagram

![LOGSCAN syntax diagram]

Figure 38: LOGSCAN syntax—Table space specification

![Table space specification diagram]
Figure 39: LOGSCAN syntax—Single index specification
Figure 40: LOGSCAN syntax—Multiple index specification

Multiple index specification

INDEX

INDEXSPACE

TABLESPACE

TABLESPACE

OBJECTSET

creator.name.

creatorID.

indexName.

tableSpaceName.

INDEXSPACE

databaseName.

indexSpaceName.

INDEX

databaseName.

tableSpaceName.

AUX

NO

YES

HISTORY

ARCHIVE

XML

LOB

AUX

NO

YES

HISTORY

ARCHIVE

XML

LOB

Figure 41: LOGSCAN syntax—TORBA or TOLOGPOINT specification

TORBA or TOLOGPOINT specification

TORBA

TOLOGPOINT

LASTQUIESCE

LASTCOMMONQ

LASTARCHQ

LASTSHUTDOWN

X’logPoint’

(relativeGenerationNumber)

BACKOUT

YES
LOGSCAN command dependencies and prohibitions

The dependencies and prohibitions for using the LOGSCAN command are as follows:

- A LOGSCAN command cannot be included in the same SYSIN with any of the following statements:
  - RECOVER TABLESPACE
  - RECOVER INDEXSPACE
  - RECOVER INDEX
  - RECOVER UNLOADKEYS
  - RECOVER BUILDINDEX
  - REBUILD INDEX
  - ACCUM

- If you specify FROMRBA or FROMLOGPOINT, you must specify LOGAPPLY ONLY as well.

- If you specify BACKOUT, you cannot specify LOGONLY or LOGAPPLY ONLY.

- The LOGSCAN command does not consider change accumulation files.

Additionally, consider the following items when you scan indexes:

- If you have used the BMC PACLOG for DB2 utility to exclude the index log records, the LOGSCAN command cannot be used for such indexes.

- If you specify LOGSCAN INDEX (ALL) or LOGSCAN INDEXSPACE (ALL), LOGSCAN INDEX and LOGSCAN INDEXSPACE do not support the FROMRBA/FROMLOGPOINT options. You can use separate index specifications with LOGSCAN INDEX and LOGSCAN INDEXSPACE to use those options.

- If a table space is scanned to a prior point in time, the default TORBA or TOLOGPOINT value for recovery of any indexes on that table space is the log point specified for the table space. If you specify different log points for the table space and its index with TORBA or TOLOGPOINT, NGT Recover ends with an error.

- NGT Recover does not prevent a scan a specific point in time for an index that does not include a scan for the table space or other indexes on the table space.

- Indexes must always have a copy or other backup if LOGSCAN INDEX or LOGSCAN INDEXSPACE is used without BACKOUT. If an index is created on a table that contains data, DB2 does not log the index updates. Similarly, index rebuilds or updates that result from LOG YES utilities are not logged. If the index is created on a table, an image copy or other backup must be made before the index can be scanned.
In addition to the LOG NO SYSCOPY events that make a table space unrecoverable, any LOAD LOG YES or REORG LOG YES event between the image copy and the TORBA or TOLOGPOINT value makes an index unrecoverable. REBUILD INDEX and REORG INDEX can also render an index unrecoverable, and NGT Recover cannot detect these events if non-BMC utilities are used.

You cannot use LOGSCAN INDEX or LOGSCAN INDEXSPACE through a point-in-time recovery event created by using BACKOUT.

**LOGSCAN option descriptions**

LOGSCAN provides information about the number and size of log records required for recovery. You can use this information to size the log sort. A report is printed describing the log records for each object named in the LOGSCAN.

This section describes the options available with LOGSCAN. Not all of the options described are valid in a single LOGSCAN TABLESPACE, LOGSCAN INDEX, or LOGSCAN INDEXSPACE command. The recovery strategy that best meets your needs determines which options you should include in a single command statement.

The LOGSCAN TABLESPACE, LOGSCAN INDEX, and LOGSCAN INDEXSPACE commands specify one or more table spaces or indexes, or table space or index data sets, for scanning. A single table space, index, or data set can be scanned to a specific log point or to the current state (by using all of the log records that apply). Multiple table spaces, indexes, or data sets can also be scanned to a specific log point or to the current state. You can have as many LOGSCAN TABLESPACE, LOGSCAN INDEX, or LOGSCAN INDEXSPACE commands in a job step as necessary; however, NGT Recover examines all command statements in the job step before processing starts so that all log processing can be combined and activities can be optimally scheduled.

**Table space specification**

The syntax diagram for the Table Space specification is in "LOGSCAN syntax" on page 254.

**TABLESPACE databaseName.tableSpaceName**

The TABLESPACE option specifies the table space to be scanned as follows:

- *databaseName* is the name of the database containing the table space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database that is defined with TYPE='W' in SYSIBM.SYSDATABASE.
- `tableSpaceName` is the name of the table space within the named database. BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

**OBJECTSET**

Use the OBJECTSET option with TABLESPACE to process only the table spaces that are in a defined group. For more information, see “OBJECTSET specification” on page 265 and “Using BMC RECOVERY MANAGER groups” on page 518.

---

**Note**

To include the indexes in a log scan using TABLESPACE OBJECTSET, specify INDEXES YES.

---

**INDEXES**

The INDEXES option allows you to specify that you want NGT Recover to process the indexes associated with the table space(s) given by the TABLESPACE option of the LOGSCAN command. The default is INDEXES NO indicating no indexes processed.

---

**Note**

The use of INDEX is synonymous to INDEXES for this option.
The INDEXES option is not applicable to INDEXSPACE or INDEX specifications.

---

**INDEXES NO**

Specifying INDEXES NO tells NGT Recover to not process the indexes for the specified table space or table spaces.

**INDEXES YES**

Specifying INDEXES YES tells NGT Recover to process of all indexes associated with the table space(s) specified by the TABLESPACE.

---

**Note**

INDEXES YES is invalid with an unqualified OBJECTSET specification (OBJECTSET without TABLESPACE).

---

**DSNUM**

DSNUM specifies a single data set in the specified table space or the entire table space.
DSNUM ALL

DSNUM ALL is the default and specifies that all data sets in the table space are to be scanned. If the table space is nonpartitioned and DSNUM ALL is specified or implied, image copies that are made for specific data sets will be considered only if they are registered at the same log point.

DSNUM integer

Use this option to scan only one partition or data set in the table space. For partitioned table spaces, DSNUM integer specifies the number (from 1 through 4096) of a single partition in the specified table space. For nonpartitioned table spaces, DSNUM integer specifies the number (from 1 to 32) of a single data set of the specified table space.

DSNUM begin : end

Use this option to recover a range of partitions in the table space. For partitioned table spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

AUX

The AUX option specifies if auxiliary objects will be included with the scan of the base table spaces. For more information, see “AUX” on page 117.

FROMRBA or FROMLOGPOINT

You can specify FROMRBA or FROMLOGPOINT only with the LOGAPPLY ONLY option. When you use LOGAPPLY ONLY, you must specify FROMRBA or FROMLOGPOINT for each table space and index specification.

FROMRBA or FROMLOGPOINT specifies the log point in the log where a scan of the log records should begin for the object or object data set being scanned. You can specify the log point with a keyword or with a hexadecimal string.

This option assumes that the object or objects have been brought to an appropriate point by some other activity prior to this step.

You can use the FROMRBA and FROMLOGPOINT keywords interchangeably, regardless of the version of DB2 that you are using.

FROMRBA LASTQUIESCE or FROMLOGPOINT LASTQUIESCE

FROMRBA LASTQUIESCE or FROMLOGPOINT LASTQUIESCE specifies that the log point where a scan of the log records will begin for this object (or
object data set) is the log point of the most recent quiesce point registered in SYSLIBM.SYSCOPY. If this is recovery of an index, the QUIESCE entry for the table space is used. You can abbreviate LASTQUIESCE to LASTQ.

**Note**

If you use LASTQUIESCE and the quiesce is prior to the ALTER ADD PART, the recovery will fail. You must specify a hard coded RBA. If the partition was added for a partition-by-growth universal table space, you can use the LASTQUIESCE option.

FROMRBA LASTCOPY or FROMLOGPOINT LASTCOPY

FROMRBA LASTCOPY or FROMLOGPOINT LASTCOPY specifies that the log point where a scan of the log records will begin for this object (or object data set) is the log point of the most recent image copy (full or incremental) registered.

FROMRBA LASTARCHQ or FROMLOGPOINT LASTARCHQ

FROMRBA LASTARCHQ or FROMLOGPOINT LASTARCHQ specifies that the log point of the most recent ARCHIVE LOG MODE(QUIESCE) command is the point where a scan of the log records will begin for this object (or object data set).

FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN

FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN specifies that the log point of the most recent successful STOP DB2 command is the point where a scan of the log records will begin for this object (or object data set).

**Note**

If you specify this keyword in a data sharing environment, log records since the last successful shutdown of the member subsystem on which the job is running are used to scan, without regard to other member subsystems. For this reason, use extreme caution when you use FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN in a data sharing environment to ensure that the shutdown used is really a point consistent with the state of the spaces.

FROMRBA X'logPoint' or FROMLOGPOINT X'logPoint'

FROMRBA X'logPoint' or FROMLOGPOINT X'logPoint' specifies the log point in the log where a scan of the log records for this object will begin.

'logPoint' is a string of up to twelve hexadecimal digits.
Single index specification

The syntax diagram for the Single index specification is in “LOGSCAN syntax” on page 254.

INDEX creatorID.indexName

The INDEX option specifies the index to be scanned as follows:

- creatorID is the qualifier creator ID for the index. The default is the user identifier for the utility.

- indexName is the name of the index. Indexes for BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

OBJECTSET creator.name

Use the OBJECTSET option with INDEX to process only the indexes that are in a defined group. For more information, see “OBJECTSET specification” on page 265 and “Using BMC RECOVERY MANAGER groups” on page 518.

INDEXSPACE databaseName.indexSpaceName

The INDEXSPACE option specifies the index space to be scanned.

- databaseName is the name of the database containing the index space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE.

- indexSpaceName is the name of the index space within the named database. Indexes for BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

DSNUM

DSNUM specifies either a single data set for the specified index or the entire index.

DSNUM ALL

DSNUM ALL is the default and specifies that all of the data sets for the index are to be scanned.

If the index is nonpartitioned and DSNUM ALL is specified or implied, image copies made for specific data set will be considered only if they are registered at the same log point.
For multi-data-set, nonpartitioned index spaces, if the index copy used is a DSN1COPY, DSNUM ALL is not allowed.

**DSNUM integer**

Use this option to scan only one partition or data set for an index. For partitioned indexes, DSNUM integer specifies the number (from 1 through 4096) of a single partition of the specified index. For nonpartitioned indexes, DSNUM integer specifies the number (from 1 to 4096) of a single data set for the specified index.

**DSNUM begin : end**

Use this option to recover a range of partitions in the table space. For partitioned table spaces, begin specifies the number (from 1 through 4095) of the first partition in the range and end specifies the number (from 2 through 4096) of the last partition in the range. The two numbers are separated by a colon (:) with or without spaces. Wrapping partition numbers (for example, DSNUM 4050 : 300) is not supported.

**FROMRBA or FROMLOGPOINT**

For a description of FROMRBA and FROMLOGPOINT and their options, see “Table space specification” on page 258.

### Multiple index specification

The syntax diagram for the Multiple index specification is in “LOGSCAN syntax” on page 254.

**INDEX creatorID.indexName**

The INDEX option specifies the index to be scanned.

- *creatorID* is the qualifier creator ID for the index. The default is the user identifier for the utility.
- *indexName* is the name of the index. Indexes for BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

**OBJECTSET creator.name**

Use the OBJECTSET option with INDEX to process only the indexes that are in a defined group. For more information, see “OBJECTSET specification” on page 265 and “Using BMC RECOVERY MANAGER groups” on page 518.
INDEX (ALL) TABLESPACE `databaseName.tableSpaceName`

(ALL) TABLESPACE specifies that all indexes for the named table space are to be scanned. The parentheses around ALL are optional.

TABLESPACE specifies the table space from which all indexes are to be scanned as follows:

- `databaseName` is the name of the database containing the table space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE.

- `tableSpaceName` is the name of the table space within the named database. BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

AUX

The AUX option specifies if auxiliary objects will be included with the scan of the base table spaces. For more information, see "AUX" on page 117.

INDEXSPACE `databaseName.indexSpaceName`

The INDEXSPACE option specifies indexes to be scanned as follows:

- `databaseName` is the name of the database containing the index space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE.

- `indexSpaceName` is the name of the index space within the named database. Indexes for BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

INDEXSPACE (ALL) TABLESPACE `databaseName.tableSpaceName`

INDEXSPACE (ALL) TABLESPACE specifies that all indexes for the named table space are to be scanned.

TABLESPACE specifies the table space from which all indexes are to be scanned as follows:

- `databaseName` is the name of the database to which the table space belongs. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database defined with TYPE='W' in SYSIBM.SYSDATABASE. The default is DSNDB04.
**tableSpaceName** is the name of the table space whose indexes are to be scanned. BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

If you do not specify the TABLESPACE option, NGT Recover determines the table space for the first valid index specified. All other specified indexes must belong to the same table space.

## CLONE

The CLONE option indicates that NGT Recover is to scan clone data. When you work with clone data, the following limitations apply:

- You cannot refer to the same object with the CLONE option and without the CLONE option in the same recovery step; you cannot process the base and its clone in the same command.

- Related objects (table space and all indexes for its table, all related LOB objects) should use the same CLONE specification in the same recovery step.

## OBJECTSET specification

The OBJECTSET option is available in LOGSCAN statement syntax to process logs for groups that are defined using RECOVERY MANAGER. OBJECTSET is available in the following syntactical forms.

The OBJECTSET specification is shown in “LOGSCAN syntax” on page 254

- **LOGSCAN OBJECTSET creator.name**  
  LOGSCAN OBJECTSET performs log scan processing for all objects in a group, both table spaces and indexes.

  _Note_  
  INDEXES YES is not valid with a LOGSCAN OBJECTSET specification.

- **LOGSCAN TABLESPACE OBJECTSET creator.name**  
  LOGSCAN TABLESPACE OBJECTSET performs log scan processing only for the table spaces in the group.

- **LOGSCAN TABLESPACE OBJECTSET creator.name INDEXES YES**  
  LOGSCAN TABLESPACE OBJECTSET... INDEXES YES performs log scan processing for the table spaces in the group along with their associated indexes regardless of whether the indexes are included in the group.
- **LOGSCAN INDEX OBJECTSET** *creator.name*

  LOGSCAN TABLESPACE OBJECTSET performs log scan processing only for the indexes in the group.

The following rules apply to *creator.name*:

- *creator* specifies the name of the creator of the group and can have up to 128 characters in length.
- *name* specifies the group name and can have up to 128 characters in length.
- You can delimit both *creator* and *name* with single or double quotation marks.
- Both *creator* and *name* can contain the special characters $, #, and @ in any position.

NGT Recover uses dynamic grouping to determine the group contents at the time that you run the NGT Recover job containing the OBJECTSET option. Because NGT Recover reads the objects in the group each time the job is executed, objects may be added or removed from the group.

**Note**

NGT Recover does not read RECOVERY MANAGER recovery options for the group. If RECOVERY MANAGER recovery options change, you must run RECOVERY MANAGER to pick up the new options and generate the control cards.

Following are restrictions that apply when you use OBJECTSET in any of its forms on the LOGSCAN statement:

- AUX is ignored with any specification of OBJECTSET.
- You cannot specify DSNUM with OBJECTSET. NGT Recover uses the group definition for the object.
- NGT Recover allows only one OBJECTSET clause, and the statement cannot contain additional TABLESPACE, INDEX, or INDEXSPACE clauses.

**TORBA or TOLOGPOINT specification**

The following section describes the TORBA or TOLOGPOINT specification shown in “LOGSCAN syntax” on page 254.
TORBA or TOLOGPOINT

Use TORBA or TOLOGPOINT to scan a table space or index to a prior point in time identified by a log point. (For more information about using TORBA or TOLOGPOINT with table spaces and indexes, see LOGSCAN command dependencies and prohibitions on page 257.)

You can use TORBA or TOLOGPOINT interchangeably, regardless of the version of DB2 that you are using.

TORBA LASTQUIESCE (relativeGenerationNumber) or TOLOGPOINT LASTQUIESCE (relativeGenerationNumber)

Use TORBA LASTQUIESCE or TOLOGPOINT LASTQUIESCE to scan to the most recent quiesce point registered in SYSIBM.SYSCOPY for the table space being scanned.

For indexes, the most recent quiesce of the table space is used. NGT Recover scans log records with starting log points less than or equal to the most recent quiesce point, unless BACKOUT is used. For BACKOUT requests, NGT Recover scans log records with starting log points greater than the most recent quiesce point.

Optionally, you can add a relative generation number in parentheses to indicate the quiesce to use. A relative generation number of (0) uses the most recent image copy. For example, TOLOGPOINT LASTQUIESCE (-1) indicates the use of the quiesce before the most recent quiesce.

If you are scanning multiple table spaces in a single LOGSCAN TABLESPACE command statement, all of the table spaces must have a common last quiesce point. However, if you are using multiple LOGSCAN TABLESPACE command statements to scan multiple table spaces, each table space can have a different last quiesce point. The same is true for indexes if you are using LOGSCAN INDEX or LOGSCAN INDEXSPACE.

LASTQUIESCE can be abbreviated as LASTQ.

Note

If you use LASTQUIESCE and the quiesce is prior to the ALTER ADD PART, the recovery will fail. You must specify a hard-coded RBA. If the partition was added for a partition-by-growth universal table space, you can use the LASTQUIESCE option.

TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ

Use TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ to scan a set of table spaces and indexes to the most recent common quiesce point registered in SYSIBM.SYSCOPY. The scanning of indexes is based on the common quiesce point of the owning table space. NGT Recover scans log
records with starting log points less than or equal to the most recent quiesce point common to all table spaces and indexes listed, unless BACKOUT is used. For BACKOUT requests, NGT Recover scans log records with starting log points greater than the most recent common quiesce point. If no common quiesce point exists for all table spaces and indexes specified, an error message is issued.

If you specify this option when only one table space or index is scanned, the effect is exactly the same as using TORBA LASTQUIESCE.

You can use TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ with LOGSCAN INDEX (ALL) or when multiple index space specifications exist in one command statement.

TORBA LASTARCHQ or TOLOGPOINT LASTARCHQ

Use TORBA LASTARCHQ or TOLOGPOINT LASTARCHQ to scan to the log point of the last ARCHIVE LOG MODE(QUIESCE) command issued for the DB2 subsystem. NGT Recover scans only log records with starting log points less than or equal to the most recent ARCHIVE LOG MODE(QUIESCE) command, unless BACKOUT is used. For BACKOUT requests, NGT Recover scans log records with starting log points greater than the most recent ARCHIVE LOG MODE(QUIESCE) command.

When you use this option, NGT Recover issues a message indicating the date and time of the ARCHIVE LOG MODE(QUIESCE) command used.

TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN

Use TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN to scan to the log point of the last normal system shutdown. NGT Recover scans only log records with starting log points less than or equal to the most recent successful subsystem shutdown, unless BACKOUT is used. For BACKOUT requests, NGT Recover scans log records with starting log points greater than the most recent successful subsystem shutdown.

When you use this option, NGT Recover issues a message indicating the date and time of the last successful system shutdown. You cannot use the LASTSHUTDOWN keyword to scan to the log point of an abnormal DB2 termination. When you use this keyword, ensure that the table spaces and indexes were not in an exception status at the shutdown.
**Note**

In a data sharing environment, this keyword is used to scan to the last successful shutdown of the member subsystem on which the job is running, without regard to other member subsystems. For this reason, use extreme caution when you use TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN in a data sharing environment to ensure that the shutdown used is really a point of data consistency. To ensure that the last successful shutdown is really a point of data consistency, one of the following conditions must be true:

- The subsystem on which the NGT Recover job is running was the last member of a data sharing group to be stopped.
- All updates to the table spaces and indexes to be scanned occurred on the subsystem on which the NGT Recover job is running.

**TORBA X'logPoint' or TOLOGPOINT X'logPoint'**

Use TORBA X'logPoint' or TOLOGPOINT X'logPoint' to scan to a prior point in time identified by the log point, 'logPoint'. RECOVERY MANAGER scans only log records with starting log points less than or equal to 'logPoint', unless BACKOUT is used. For BACKOUT requests, NGT Recover scans log records with starting log points greater than 'logPoint'.

'logPoint' is a string of up to twelve hexadecimal digits.

**BACKOUT**

BACKOUT invokes the backout strategy for point-in-time recovery by using log points (TORBA, TOLOGPOINT). This strategy assumes that spaces are undamaged and that you require a reset to a specific point in time. The spaces are used with the log records between the point in time and the current point to back out to the required state. Using BACKOUT may enhance the performance of a point-in-time LOGSCAN request significantly.

The following restrictions apply:

- The space must be current as of the last logged activity and not damaged in any way. Multiple-data set, nonpartitioned spaces must have all data sets scanned (DSNUM ALL).
- No LOAD or REORG events can exist between the log point the specified and the current log points. For indexes, no REBUILD INDEX events can exist in this range.
- No prior point-in-time recovery with a START_RBA greater than the log point requested and a PIT_RBA less than the log point requested can exist.
- BACKOUT may not be requested with LOGONLY or LOGAPPLY ONLY.
**Note**
BACKOUT uses only logs and spaces but requires that the spaces be current. The LOGONLY and LOGAPPLY ONLY options imply scanning log records going forward by using a space restored to a previous state.

- Change accumulation files are not allowed with BACKOUT because they are not properly ordered. Output accumulation files are also not supported because they are defined from the point of the last image copy to the current point.

**LOGAPPLY ONLY**

LOGAPPLY ONLY specifies that only log information is to be scanned for a table space or index and that no image copy data sets are to be used. Use this option with FROMRBA or FROMLOGPOINT.

When you want to scan only log records for multiple table spaces or indexes in a single LOGSCAN TABLESPACE, LOGSCAN INDEX, or LOGSCAN INDEXSPACE command, you need to specify only a single LOGAPPLY ONLY. However, you must specify FROMRBA or FROMLOGPOINT for each table space or index.

LOGAPPLY ONLY is not valid with BACKOUT.

**LOGONLY**

LOGONLY scans only the log records to the data sets. All log records written after a point recorded in the data set are used. Specify LOGONLY when the data sets of the target objects have been restored by using another process and the HPGRBRBA value in the header pages is correctly set to indicate where log apply should begin. You can use this option on single table spaces or indexes, or on table space or index lists.

LOGONLY is not valid with BACKOUT.

**MIGRATE**

The MIGRATE command allows you to move data from a one or more table spaces to another. The table spaces can be on the same subsystem, or they can be on different subsystems, as long as the two subsystems share DASD.

A migration file that the NGT Copy EXPORT command creates facilitates the migration. You run the EXPORT command on the target subsystem to collect information on one or more target table spaces. This information includes object identifiers and other information relating to the target table spaces. You run the NGT Recover MIGRATE command on the source subsystem to migrate all the table spaces.
to the target table spaces that are identified by the command. The INDEP OUTSPACE specification is required in order to specify the target VSAM data set model name. Prior to executing the MIGRATE command, the target table spaces must be stopped.

**Note**

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

The MIGRATE command queries the source SYSCOPY and BMCXCOPY tables for the required image copy, and then merges the copy with log (if requested) and translates object identifiers to create the target image. The log information can come from DB2 log and change accumulation files. If you specify FROMRBA, only log is used for the migration.

**Note**

This command requires one of the following valid passwords:

- A Recovery Management solution password
- A Database Administration solution password

The Database Administration (DAD) password allows NGT Recover to perform data migration functions that were generated by CHANGE MANAGER for DB2 within a worklist, but does not include full-function NGT Recover.

For more information about copy migration using NGT Copy EXPORT and NGT Recover MIGRATE, see “Using MIGRATE and IMPORT for data movement” on page 520 and the *Recovery Management for DB2 User Guide*.

**MIGRATE syntax**

The following figures show the syntax of the MIGRATE command.
Figure 42: MIGRATE syntax

MIGRATE \textsuperscript{1} \textbf{USING specification}

\begin{itemize}
\item TABLESPACE \textbf{Name}
\item \textbf{FROMRBA or FROMLOGPOINT specification}
\item OBJECTSET specification
\item INDEPENDENT OUTSPACE specification
\item TORBA or TOLOGPOINT recovery specification
\item TOCOPY \textbf{LASTCOPY} \textbf{(relativeGenerationNumber)}
\item LOGSORT specification
\item LOGAPPLY ONLY \textbf{LOGONLY} \textbf{CLONE} \textbf{ANALYZE} \textbf{YES} \textbf{NO} \textbf{ONLY} \textbf{TOLOCATION location}
\end{itemize}

\textsuperscript{1} Requires a valid password as follows: Recovery Management or Database Administration
Figure 43: MIGRATE syntax—USING specification

**USING specification**

```
USING dataSetName
```

```
CHANGE TABLE FROM creator1.tableName1 TO creator2.tableName2
```

```
CHANGE INDEX FROM creator1.indexName1 TO creator2.indexName2
```

```
INDEXES
RECOVER
NO
```

```
EXCLUDE (dbname.ts qualifier.ix)
```

```
DSNUM
ALL
integer
begin : end
```

```
RESET
```

Figure 44: MIGRATE syntax—FROMRBA or FROMLOGPOINT specification

**FROMRBA or FROMLOGPOINT specification**

```
FROMRBA
FROMLOGPOINT
```

```
LASTQUIESCE
LASTCOPY
LASTARCHQ
LASTSHUTDOWN
X'logPoint'
```
Figure 45: MIGRATE syntax—OBJECTSET specification

OBJECTSET specification

```plaintext
OBJECTSET creator.name

EXCLUDE (dbname.ts qualifier.ix )
```

Figure 46: MIGRATE syntax—INDEPENDENT OUTSPACE specification

INDEPENDENT OUTSPACE specification

```plaintext
INDEP OUTSPACE

MODEL vcat.BMCDBC.databaseName.objectName.10001.zznn

MODEL userNamedDataSet
```

Figure 47: MIGRATE syntax—TORBA or TOLOGPOINT specification

TORBA or TOLOGPOINT specification

```plaintext
TORBA TOLOGPOINT

LASTQUIESCE

LASTCOMMUNQ

LASTARCHQ

LASTSHUTDOWN

LOGMARK logMarkName

X'logPoint'

(relativeGenerationNumber)

(logMarkGeneration)
```
Figure 48: MIGRATE syntax—LOGSORT specification

LOGSORT specification

MIGRATE command dependencies and prohibitions

The MIGRATE command has the following limitations:

- No support for copies of the DB2 catalog

- No support for specifying DSNUM
  When selecting image copies, the MIGRATE command uses a DSNUM 0 copy or DSNUM n copies. DSNUM n copies will only be considered when all the partitions or data sets are registered at the same log point.

- No support for rebuilding migrated indexes
  However, NGT Recover can recover any indexes in the migration data set depending on the value of the INDEXES parameter in the USING clause. If the source and target table spaces or index spaces are not at the same version, you will need to run REPAIR VERSIONS on the target system after the MIGRATE is executed. If the indexes for a versioned space were not migrated, then these indexes would need to be rebuilt in a separate step on the target system.

- No support for migrating LOB or XML spaces
  NGT Recover cannot migrate LOB or XML spaces when the base table space is a partition-by-growth (PBG) space and there are fewer partitions on the target than on the source. In this case, adding partitions to the base table space on the target system causes NGT Recover to create new auxiliary table spaces, but these new table spaces will not be present in the migration file.
The following table describes the common dependencies for coding the MIGRATE command.

### Table 7: MIGRATE command dependencies

<table>
<thead>
<tr>
<th>Option coded</th>
<th>Option that must be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROMRBA or FROMLOGPOINT</td>
<td>LOGAPPLY ONLY</td>
</tr>
<tr>
<td>BACKOUT</td>
<td>TORBA or TOLOGPOINT</td>
</tr>
</tbody>
</table>

The following table describes the common prohibitions to coding the MIGRATE commands.

### Table 8: MIGRATE command prohibitions

<table>
<thead>
<tr>
<th>Option coded</th>
<th>Option that must not be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGONLY</td>
<td>TOCOPY</td>
</tr>
</tbody>
</table>

### MIGRATE option descriptions

This section describes the options available with MIGRATE.

Not all of the options described are valid in a single MIGRATE command. The recovery strategy that best meets your needs determines which options you should include in a single command statement.

**MIGRATE**

Use the MIGRATE command when you want to perform a migration.

**USING specification**

This topic describes the USING specification.

The syntax diagram for the USING specification is in “MIGRATE syntax” on page 271.

**USING dataSetName**

The USING option is required when you specify the MIGRATE command. USING specifies the name of the migration file created by the NGT Copy EXPORT command. The migration file contains migration information for one or more table
spaces. This information includes object identifiers, table space and index space names, and other data relating to the target table spaces and index spaces.

**CHANGE TABLE FROM creator1.tableName1 TO creator2.tableName2 or CHANGE INDEX FROM creator1.indexName1 TO creator2.indexName2**

When the table names or index names are not the same between the source and the target subsystems, use the CHANGE ... FROM ... TO syntax to resolve differences.

Each table or index must contain a creator and a name. You can use an asterisk (*) as any portion of the name.

You must take care when using wildcarding. If the name in the TO clause does not produce a one-to-one match for all the tables or indexes on the target space, the space is considered unrecoverable. (The FROM name is the name in the migration file. For MIGRATE, that is the name on the target system. The TO name is the name on the source for MIGRATE.)

For example, you may want to copy the production payroll tables to the test subsystem, but the creator names between these two subsystems are different. Use the CHANGE TABLE feature to identify table name matches between the two subsystems. The migration file is searched for the FROM table name pattern and is substituted with the TO table name pattern. Multiple CHANGE ... FROM ... TO clauses are allowed in a single USING specification.

In the following specification, all the tables in the migration file with the creator name of PAYTEST resolve to a creator name of PAYPROD, which matches the tables on the source subsystem.

**MIGRATE USING dataSetName CHANGE TABLE FROM PAYTEST.* TO PAYPROD.***

**INDEXES**

The INDEXES option specifies how NGT Recover handles indexes when you specify the MIGRATE command. (You can also use INDEX or INDEXS in place of INDEXES.)

**INDEXES RECOVER**

If there are indexes in the migration data set, they will be recovered.

**INDEXES NO**

INDEXES NO specifies that NGT Recover does not recover indexes that are present in the migration data set.
EXCLUDE (dbname.ts or qualifier.ix)

Use the EXCLUDE option in the USING specification to exclude one or more objects from migration. You can use wildcards or specific names to specify the exclusions. The following wildcards are valid:

- % and * match one or more characters in the object name.
- _ and ? match a single character in the object name.

List the excluded objects following the EXCLUDE keyword. Each item in the list must be in the form dbname.ts or qualifier.ix and you must separate the individual items with commas. Enclose the list in parentheses.

DSNUM

You can exclude multiple partitions from the specified object.

- DSNUM ALL is allowed with partitioned and nonpartitioned objects.
- DSNUM begin: end is allowed for partitioned objects to exclude a range of partitions.
- DSNUM integer is allowed for partitioned objects to exclude a single partition.

RESET

The RESET keyword causes the log points in each data page to be reset to '0'. You must specify RESET when you are migrating data from one DB2 non-data-sharing system to another because the log point values on the first subsystem are meaningless or misleading on the target DB2.

STATS

The STATS option allows you to carry forward existing RUNSTATS from a migration file. These RUNSTATS may then be optionally applied during an NGT Recover IMPORT. To import stats, you must have specified STATS YES in the NGT Copy EXPORT command.

SYSINDEXSTATS, SYSLOBSTATS, and SYSTABSTATS are included in the migration file. Real-time statistics are included in a migration, regardless of the STATS specification.

STATS YES

STATS YES updates RUNSTATS information on the target system using information from the migration file. RTS stats are also updated.
STATS NO

RUNSTATS information is not updated. RTS stats are always updated. STATS NO is the default.

TABLESPACE specification

The syntax for the TABLESPACE specification is in Figure 42 on page 272.

TABLESPACE databaseName.tableSpaceName

The TABLESPACE option specifies the source table space to be migrated.

- databaseName is the name of the database containing the table space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database that is defined with TYPE='W' in SYSIBM.SYSDATABASE.

- tableSpaceName is the name of the table space within the named database. BMCUTIL, BMCSYNC, and BMCXCOPY (tables that are used by BMC utilities) cannot be used.

FROMRBA or FROMLOGPOINT specification

You can specify FROMRBA or FROMLOGPOINT only with LOGAPPLY ONLY. When you use LOGAPPLY ONLY, you must specify FROMRBA or FROMLOGPOINT for each table space specification.

FROMRBA or FROMLOGPOINT specifies the log point in the log where application of the log records should begin for the object or object data set being migrated. You can specify the log point with a keyword or with a hexadecimal string.

This option assumes that the object or objects have been brought to an appropriate point by some other activity prior to this step.

You can use the FROMRBA and FROMLOGPOINT keywords interchangeably, regardless of the version of DB2 that you are using.

The syntax diagram for the FROMRBA or FROMLOGPOINT specification is in “MIGRATE syntax” on page 271.

FROMRBA LASTQUIESCE or FROMLOGPOINT LASTQUIESCE

FROMRBA LASTQUIESCE or FROMLOGPOINT LASTQUIESCE specifies that the log point where application of the log records will begin for this
object (or object data set) is the log point of the most recent quiesce point registered in SYSIBM.SYSCOPY.

**Note**
If you use LASTQUIESCE and the quiesce is prior to the ALTER ADD PART, the migration will fail. You must specify a hard-coded RBA. If the partition was added for a partition-by-growth universal table space, you can use the LASTQUIESCE option.

**FROMRBA LASTCOPY or FROMLOGPOINT LASTCOPY**

FROMRBA LASTCOPY or FROMLOGPOINT LASTCOPY specifies that the log point where application of the log records will begin for this object (or object data set) is the log point of the most recent image copy (full or incremental) registered.

**FROMRBA LASTARCHQ or FROMLOGPOINT LASTARCHQ**

FROMRBA LASTARCHQ or FROMLOGPOINT LASTARCHQ specifies that the log point of the most recent ARCHIVE LOG MODE(QUIESCE) command is the point where application of the log records will begin for this object (or object data set).

**FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN**

FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN specifies that the log point of the most recent successful STOP DB2 command is the point where application of the log records will begin for this object (or object data set).

**Note**
If you specify this keyword in a data sharing environment, log records since the last successful shutdown of the member subsystem on which the job is running are used for migration, without regard to other member subsystems. For this reason, use extreme caution when you use FROMRBA LASTSHUTDOWN or FROMLOGPOINT LASTSHUTDOWN in a data sharing environment to ensure that the shutdown used is really a point consistent with the state of the spaces.

**FROMRBA X'logPoint' or FROMLOGPOINT X'logPoint'**

FROMRBA X'logPoint' or FROMLOGPOINT X'logPoint' specifies the log point in the log where application of the log records for this object will begin.

'logPoint' is a string of up to twelve hexadecimal digits.
OBJECTSET specification

The OBJECTSET option is available in MIGRATE statement syntax to migrate groups that are defined using RECOVERY MANAGER.

The OBJECTSET specification is shown in “MIGRATE syntax” on page 271.

OBJECTSET creator.name

The following rules apply to creator.name:

- creator specifies the name of the creator of the group and can have up to 128 characters in length.
- name species the group name and can have up to 128 characters in length.
- You can delimit both creator and name with single or double quotation marks.
- Both creator and name can contain the special characters $, #, and @ in any position.

NGT Recover uses dynamic grouping to determine the group contents at the time that you run the NGT Recover job containing the OBJECTSET option. Because NGT Recover reads the objects in the group each time the job is executed, objects may be added or removed from the group.

Following are restrictions that apply when you use OBJECTSET in any of its forms on the RECOVER statement:

- You cannot specify DSNUM with OBJECTSET. NGT Recover uses the group definition for the object.
- NGT Recover allows only one OBJECTSET clause, and the statement cannot contain additional TABLESPACE clauses.

When you use OBJECTSET, if NGT Recover tries to migrate a group of objects and more than 10 of the objects are unrecoverable, NGT Recover issues an error for each object that is unrecoverable and discontinues further processing of the group. This behavior is based on the ERRCONT option that has a default value of 10. To avoid this behavior, set the ON ERROR CONTINUE option with a value that will exceed the possible number of errors in the group, and the migration will complete.

If a table space is found in the OBJECTSET but it is not found in the migration file, the table space is considered unrecoverable. You can use the EXCLUDE specification to eliminate this object from the OBJECTSET.
EXCLUDE (dbname.ts or qualifier.ix)

Use the EXCLUDE option after an OBJECTSET specification to exclude one or more objects from the migration. You can use wildcards or specific names to specify the exclusions. The following wildcards are valid:

- % and * match one or more characters in the object name.
- _ and ? match a single character in the object name.

List the excluded objects following the EXCLUDE keyword. Each item in the list must be in the form dbname.ts or qualifier.ix and you must separate the individual items with commas. Enclose the list in parentheses.

INDEPENDENT OUTSPACE specification

Use the INDEPENDENT (or INDEP) OUTSPACE option to redirect the output of the migration to a data set that is not used by the source DB2.

The syntax diagram for the INDEPENDENT OUTSPACE specification is in “MIGRATE syntax” on page 271.

INDEPENDENT OUTSPACE or INDEP OUTSPACE

Use the INDEPENDENT (or INDEP) OUTSPACE option to redirect the output of the migration to a data set that is not used by the source DB2. INDEP OUTSPACE migrations do not require you to stop the source spaces that are being migrated, and they do not interfere with production processing. They do require that you stop the target DB2 spaces during migration. INDEP OUTSPACE enables you to do the following things:

If you use TABLESPACE keyword, you can specify the name of each independent data set or you can use a model name to represent the VSAM data set names of several target table spaces.

Note

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

You can preallocate the INDEP OUTSPACE data sets using the following guidelines:

- For VCAT-defined spaces, ensure that the volumes or space allocated is large enough to hold the data set.
If the target space is defined using a STOGROUP, and if primary and secondary quantities are not defined for the source space, NGT Recover uses a sliding-scale calculation for secondary extents, similar to the method DB2 uses.

For more information, see “Primary space allocation” on page 81 and “Secondary extents” on page 81.

You can allow NGT Recover to dynamically allocate the INDEP OUTSPACE data sets for STOGROUP-defined spaces. If the INDEP OUTSPACE data set does not exist, NGT Recover uses the attributes of the source space to dynamically allocate the target INDEP OUTSPACE data set. If the INDEP OUTSPACE data set exists, NGT Recover uses it as is (no DELETE or DEFINE executed).

NGT Recover includes support for multi-data-set, nonpartitioned spaces. For example, NGT Recover will dynamically allocate A002, A003, to A00n INDEP OUTSPACE data sets as needed. For partition-by-growth table spaces, you must ALTER add partitions to the target space for data sets that do not exist.

To have NGT Recover dynamically allocate the INDEP OUTSPACE data set on a specific volume or volumes, pass a volume list by using the STOGROUP...USEORDER specification in the OPTIONS statement (see “STOGROUP specification description” on page 112). NGT Recover statements that use STOGROUP... USEORDER are shown in the following example:

```sql
OPTIONS STOGROUP TESSTO USEORDER(NEW004,NEW005,NEW006)
```

The default model name is identical to the DB2 name of the object being recovered, but with BMCDBC substituted for DSNDBC in the CLUSTER portion of the VSAM data set:

```
vcat.BMCDBC.databaseName.objectName.p0001.znnn
```

The variables are defined as follows:

- **vcat** is the ICF catalog name.

- **databaseName.objectName** is the name of the database and object containing the data set being recovered.

- **p** corresponds to the value that is used by DB2 for this node.

- **znnn** is the data set number.

  - The letter **z** represents A, B, C, D, or E. These letters correspond to the first digit—0, 1, 2, 3, or 4—of the partition number. **nnn** represents the rest of the partition number.

  - **znnn** represents the partitions numbers as follows:
    - A001 through A999 for partitions 1 through 999
    - B000 through B999 for partitions 1000 through 1999
    - C000 through C999 for partitions 2000 through 2999
When you want to specify a name of your own choosing, use the MODEL option to specify the name of the CLUSTER portion of the VSAM data set that you want to use. The data set name that you specify may include any of the symbolic variables that are listed in the description of the MODEL keyword in the following section.

When you use INDEP OUTSPACE, you must specify it after the object specification, as shown in the syntax diagram in “MIGRATE syntax” on page 271.

**MODEL**

The MODEL option specifies the non-DB2 data set cluster name to which you want to redirect the output from the recovery of the specified object. The data set name that you specify may optionally include the symbolic variables in “Using symbolic variables” on page 494.

**TORBA or TOLOGPOINT specification**

Use TORBA or TOLOGPOINT to migrate the image of a table space or index to a prior point in time identified by a log point.

The syntax diagram for the TORBA or TOLOGPOINT specification is in “MIGRATE syntax” on page 271.

**TORBA or TOLOGPOINT**

Use TORBA or TOLOGPOINT to migrate the image of a table space or index to a prior point in time identified by a log point. (For more information about using TORBA or TOLOGPOINT with MIGRATE, see MIGRATE command dependencies and prohibitions on page 275.)

You can use TORBA or TOLOGPOINT interchangeably, regardless of the version of DB2 that you are using.

A TOLOGPOINT migration creates an image of a space to a point that reflects all activity on the space up to and including a log record at the log point specified. Any log record or activity higher than TOLOGPOINT is no longer reflected in the space.

**Note**

Use of MIGRATE with TORBA or TOLOGPOINT is not a point-in-time recovery. It is simply creation of an image as of a prior point in time.
TORBA LASTQUIESCE \((relativeGenerationNumber)\) or TOLOGPOINT LASTQUIESCE \((relativeGenerationNumber)\)

Use TORBA LASTQUIESCE or TOLOGPOINT LASTQUIESCE to migrate to the most recent quiesce point registered in SYSIBM.SYSCOPY for the table space being migrated.

Optionally, you can add a relative generation number in parentheses to indicate the quiesce to use. A relative generation number of \((0)\) uses the most recent quiesce. For example, TOLOGPOINT LASTQUIESCE \((-1)\) indicates the use of the quiesce before the most recent quiesce.

If you are migrating multiple table spaces in a single MIGRATE command statement, all of the table spaces must have a common last quiesce point. However, if you are using multiple MIGRATE command statements to migrate multiple table spaces, each table space can have a different last quiesce point.

LASTQUIESCE can be abbreviated as LASTQ.

**Note**

If you use LASTQUIESCE and the quiesce is prior to an ALTER ADD PART, the migration will fail. You must specify a hard-coded RBA. If the partition was added for a partition-by-growth universal table space, you can use the LASTQUIESCE option.

TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ

Use TORBA LASTCOMMONQ or TOLOGPOINT LASTCOMMONQ to migrate a set of table spaces and indexes to the most recent common quiesce point registered in SYSIBM.SYSCOPY. If no common quiesce point exists for all table spaces and indexes specified, an error message is issued.

If you use this option when only one table space is migrated, the effect is exactly the same as using TORBA LASTQUIESCE.

TORBA LASTARCHQ or TOLOGPOINT LASTARCHQ

Use TORBA LASTARCHQ or TOLOGPOINT LASTARCHQ to migrate to the log point of the last ARCHIVE LOG MODE(QUIESCE) command issued for the DB2 subsystem.

TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN

Use TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN to migrate to the log point of the last normal system shutdown. When you use this option, NGT Recover issues a message indicating the date and time of the last successful system shutdown. You cannot use the LASTSHUTDOWN keyword to migrate to the log point of an abnormal DB2 termination. When
you use this keyword, ensure that the table spaces and indexes were not in an exception status at the shutdown.

**Note**

In a data sharing environment, this keyword is used to migrate to the last successful shutdown of the member subsystem on which the job is running, without regard to other member subsystems. For this reason, use extreme caution when you use TORBA LASTSHUTDOWN or TOLOGPOINT LASTSHUTDOWN in a data sharing environment to ensure that the shutdown used is really a point of data consistency. To ensure that the last successful shutdown is really a point of data consistency, one of the following conditions must be true:

- The subsystem on which the NGT Recover job is running was the last member of a data sharing group to be stopped.
- All updates to the table spaces to be migrated occurred on the subsystem on which the NGT Recover job is running.

**TORBA LOGMARK** `logMarkName` (`logMarkGeneration`) or **TOLOGPOINT LOGMARK** `logMarkName` (`logMarkGeneration`)

Use TORBA LOGMARK `logMarkName` or TOLOGPOINT LOGMARK `logMarkName` to migrate to a prior point in time identified by a log mark that was created in Log Master.

Optionally, you can add a log mark generation number in parentheses, `(logMarkGeneration)`. If you do not specify a log mark generation, NGT Recover uses the most recent version of the log mark. You can use one of the following ways to specify the log mark generation:

- If you specify a log mark generation as less than or equal to zero (\(<\leq 0\)\), NGT Recover treats the generation as a relative generation. NGT Recover refers to the most recent log mark with the generation of zero (0). The previous generation of the log mark is referred to with (–1), and so on.
- If you specify a log mark generation as greater than zero (\(>0\)\), NGT Recover treats the generation as an absolute generation number and uses the specified version of the log mark.

**Note**

The way that you specify an absolute generation number for a log mark in NGT Recover is different than the way you specify the absolute generation number in Log Master.
TORBA X'logPoint' or TOLOGPOINT X'logPoint'

Use TORBA X'logPoint' or TOLOGPOINT X'logPoint' to migrate to a prior point in time identified by the log point, 'logPoint'. Only log records with starting log points less than or equal to 'logPoint' are used by NGT Recover.

'logPoint' is a string of up to twelve hexadecimal digits.

TOCOPY LASTCOPY specification

The TOCOPY option specifies the last image copy to be processed by NGT Recover to migrate the specified table space (full or incremental copies).

The syntax diagram for the TOCOPY specification is in “MIGRATE syntax” on page 271.

TOCOPY

The TOCOPY option specifies the last image copy to be processed by NGT Recover to migrate the specified table space (full or incremental copies). The migration process ends after this image copy has been applied. If you specify an incremental copy for table space migration, the prior full copy and any intervening incremental copies are also used in the migration process. The image copy specified must be registered in SYSIBM.SYSCOPY as an image copy or as a "kept" incremental copy (as specified by the BMC Next Generation Technology Copy for DB2 for z/OS product) for the specified object or object data set. When you specify TOCOPY LASTCOPY, the log is not used to apply updates that are made after the copy.

After a TOCOPY LASTCOPY migration, the data in the space reflects the data as of the time of the specified image copy. Any log record or activity after the TOCOPY LASTCOPY migration is no longer reflected in the space but may be reinstated by a subsequent TOLOGPOINT recovery.

If the copy named was created by using SHRLEVEL CHANGE, NGT Recover issues a warning (message BMC40971), but continues processing. (With a SHRLEVEL CHANGE copy, the pages may not be logically consistent.)

A point-in-time recovery event is not recorded in SYSIBM.SYSCOPY or BMCXCOPY.

TOCOPY LASTCOPY (relativeGenerationNumber)

TOCOPY LASTCOPY specifies that the most recent image copy (full or incremental for table spaces, full for indexes) registered in SYSIBM.SYSCOPY is the last one to be processed by NGT Recover to migrate the specified object.
Optionally, you can add a relative generation number in parentheses to indicate the copy to use. A relative generation number of (0) uses the most recent image copy. For example, TOCOPY LASTCOPY (-1) indicates the use of the copy before the most recent copy.

**LOGSORT specification**

For a description of the LOGSORT specification options, see

- “LOGSORT work data set specification description” on page 108

The syntax diagram for the LOGSORT specification is in “MIGRATE syntax” on page 271.

**LOGAPPLY ONLY**

LOGAPPLY ONLY specifies that only log information is to be used to migrate a table space and that no image copy data sets are to be used. Use this option when you want to specify a FROMRBA or FROMLOGPOINT.

LOGAPPLY ONLY is useful when another job has already restored one or more target table spaces to a known state. LOGAPPLY ONLY directs NGT Recover to avoid merging and restoring copies and to apply only log updates from the FROMRBA or FROMLOGPOINT specified.

When you want to use only log records to update multiple table spaces in a single MIGRATE command statement, you need to specify only a single LOGAPPLY ONLY. However, you must specify FROMRBA or FROMLOGPOINT for each table space.

When you use LOGAPPLY ONLY, you cannot use the TOCOPY LASTCOPY option.

**LOGONLY**

LOGONLY applies only the log records to the data sets. All log records written after a point recorded in the data set are applied. Specify LOGONLY when the data sets of the target objects have been restored by using another process and the HPGRBRBA value in the header pages are correctly set to indicate where log apply should begin. You can use this option on single table spaces or on multiple table spaces.

When you use LOGONLY, you cannot use the TOCOPY LASTCOPY option.

**CLONE**

The CLONE option indicates that NGT Recover is to migrate clone table data. When you migrate with clone data, the following limitations apply:
You cannot refer to the same object with the CLONE option and without the CLONE option in the same recovery step; you cannot process the base and its clone in the same command.

Related objects (table space and all indexes for its table, all related LOB objects) should use the same CLONE specification in the same recovery step.

**ANALYZE**

The ANALYZE option prints an NGT Recover plan before executing that plan.

The following information about objects is included in the plan:

- Names of any image copy data sets on which activities are or will be based
- Names of any log data sets on which activities are or will be based
- Name of any change accumulation files on which activities are or will be based
- Log ranges, if any, on which activities are or will be based
- Number of log pages to be read
- Record sizes for index sort work data sets
- Phases to occur during execution
- Steps to occur within each phase

Use this information to allocate space more accurately for log sort work data sets, index sort work data sets, and index key work data sets, thereby limiting abends that are caused by inadequate data set allocations. You can use these statistics with historical information about the times required to perform the various operations to estimate recovery time.

---

**Note**

If you coded ANALYZE with the OPTIONS command, you must code the same value here. If you code multiple NGT Recover command statements you must use the same value for the ANALYZE option in all of the command statements. If you specify a value for ANALYZE on one command statement and take the default on the others, the value that you specified on the single command statement becomes the default for the others.

---

**ANALYZE YES**

ANALYZE YES is the default and provides the information listed in the preceding section.

**ANALYZE NO**

If you specify ANALYZE NO, NGT Recover does not provide any plan information, but does provide object summaries.
ANALYZE ONLY

If you specify ANALYZE ONLY, NGT Recover provides the same information as it provides with ANALYZE YES and then stops the job when the ANALYZE phase is complete. You can use the data from the ANALYZE phase to determine the resources required and what will happen during the run. You cannot restart an NGT Recover job that specifies ANALYZE ONLY, but you can start a new job.

Use the ANALYZE ONLY option to determine which phases will occur with a specific request, or which copy data sets and log data sets will be used.

If you provide a SYSPICK DD statement, NGT Recover generates a list of all of the input tape and cartridge volumes that are allocated. For more information about SYSPICK, see “NGT Recover data sets and NGT Recover DD statements” on page 319.

If you specify OPTIONS EARLYRECALL with ANALYZE ONLY, the recall of the data sets needed for execution will be initiated. You must code OPTIONS EARLYRECALL explicitly to initiate the recalls, because OPTIONS NOEARLYRECALL is the default for ANALYZE ONLY executions.

TOLOCATION

The TOLOCATION option improves the handling of partition-by-growth spaces with more than one data set by allowing NGT Recover to automatically alter the number of data sets on the target system.

location is the location name that can be used in the CONNECT SQL statement to connect to a target subsystem.

IMPORT

The IMPORT command allows you to move data, by use of image copies, from one or more table spaces to another.

A migration file that the NGT Copy EXPORT command creates facilitates the migration. You run the EXPORT command on the source subsystem to collect information on one or more table spaces and index spaces. This information includes object identifiers, the names of image copies, and other information relating to the source table spaces and index spaces. You run the NGT Recover IMPORT command on the target subsystem to migrate all the image copy data to the target table spaces and index spaces that are identified by the command.

The IMPORT command uses the object identifiers found in the migration file and translates them to the object identifiers found in the target catalog.
Note
This command requires one of the following valid passwords:

- A Recovery Management solution password
- A Database Administration solution password

The Recovery Management (DAD) password allows NGT Recover to run within a worklist, but does not include full-function NGT Recover.

For more information about copy migration using NGT Copy EXPORT and NGT Recover IMPORT, see the *Recovery Management for DB2 User Guide*.

**IMPORT syntax**

The following figures show the syntax of the IMPORT command.
Figure 49: IMPORT syntax

1 Required a valid password as follows: Recovery Management or Database Administration
2 Not valid with BACKOUT
Figure 50: IMPORT syntax—USING specification

```
USING specification

USING dataSetName

CHANGE TABLE FROM creator1.tableName1 TO creator2.tableName2

CHANGE INDEX FROM creator1.indexName1 TO creator2.indexName2

EXCLUDE ( dbname.ts
          qualifier.ix
          ALL
          integer
          begin : end

SYNC AUTO REPLACE

INDEXES AUTO RECOVER

REBUILD NO

SYNC NOCOPYEND

STATS NO YES
```

Figure 51: IMPORT syntax—OBJECTSET specification

```
OBJECTSET creator.name

EXCLUDE ( dbname.ts
          qualifier.ix
```

Chapter 3  NGT Recover syntax  293
IMPORT command dependencies and prohibitions

Following are command dependencies and prohibitions for IMPORT:

- INCOPY is assumed and therefore is not allowed on the IMPORT command. OBID translation is also assumed.

- Since the IMPORT command requires the use of image copies, BACKOUT cannot be specified.

- When coding OUTCOPY and INDEP OUTSPACE, the output image copies cannot be registered.

- When coding INDEP OUTSPACE, REDEFINE is ignored.

- NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but
subsequently omit it from another command that supports it, NGT Recover issues an error message.

The IMPORT command has the following limitations:

- No support for incremental copies
- No support for copies of the DB2 catalog
- No support for specifying DSNUM
  When selecting image copies, the IMPORT command uses a DSNUM 0 copy or DSNUM n copies. DSNUM n copies will only be considered when all the partitions or data sets are registered at the same log point.

**IMPORT option descriptions**

This topic describes the options available with IMPORT.

Not all of the options described are valid in a single IMPORT command. The recovery strategy that best meets your needs determines which options you should include in a single command statement.

**IMPORT**

Use the IMPORT command when you want to perform a migration.

**USING specification**

USING specifies the name of the migration file created by the NGT Copy EXPORT command.

The syntax diagram for the USING specification is in “IMPORT syntax” on page 291.

**USING dataSetName**

The USING option is required when you specify IMPORT command. USING specifies the name of the migration file created by the NGT Copy EXPORT command. The migration file contains migration information for one or more table spaces and index spaces. This information includes object identifiers, copy data set names, and other data relating to the target table spaces and index spaces.
CHANGE TABLE FROM creator1.tableName1 TO creator2.tableName2 or CHANGE INDEX FROM creator1.indexName1 TO creator2.indexName2

When the table names or index names are not the same between the source and the target subsystems, use the CHANGE ... FROM ... TO syntax to resolve differences.

Each table or index must contain a creator and a name. You can use an asterisk (*) as any portion of the name.

You must take care when using wildcarding. If the name in the FROM clause does not produce a one-to-one match for all the tables or indexes on the target space, the space is considered unrecoverable.

For example, you may want to copy the production payroll tables to the test subsystem, but the creator names between these two subsystems are different. Use the CHANGE TABLE feature to identify table name matches between the two subsystems. The migration file is searched for the FROM table name pattern and is substituted with the TO table name pattern. Multiple CHANGE TABLE FROM ... TO clauses are allowed in a single USING specification.

In the following specification, all the tables in the migration file with the creator name of PAYPROD resolve to a creator name of PAYTEST, which matches the tables on the target subsystem.

```
IMPORT USING dataSetName
CHANGE TABLE FROM PAYPROD.* TO PAYTEST.*
EXCLUDE (dbname.ts or qualifier.ix)
```

Use the EXCLUDE option in the USING specification to exclude one or more objects from the import process. You can use wildcards or specific names to specify the exclusions. The following wildcards are valid:

- % and * match one or more characters in the object name.
- _ and ? match a single character in the object name.

List the excluded objects following the EXCLUDE keyword. Each item in the list must be in the form dbname.ts or qualifier.ix and you must separate the individual items with commas. Enclose the list in parentheses.

**DSNUM**

You can exclude multiple partitions from the specified object.

- DSNUM ALL is allowed with partitioned and nonpartitioned objects.
- DSNUM begin : end is allowed for partitioned objects to exclude a range of partitions.
- **DSNUM integer** is allowed for partitioned objects to exclude a single partition.

**SYNC**

Use **SYNC** to indicate if the migration should include all spaces in the IMPORT command or only those spaces that have changed since the last migration.

**SYNC** is useful in cases where the image copy indicated by the current EXPORT file is identical to the spaces imported previously.

**SYNC AUTO**

Specify **SYNC AUTO** (the default) to replace only spaces that have changed since the last time those spaces were imported. Consider this option when migrating to a query-based environment when the spaces are read only. Cost savings can be significant if you only refresh the objects that have changed since the last migration.

**SYNC REPLACE**

Specify **SYNC REPLACE** to replace all spaces regardless of whether they have changed or not. Consider this option when migrating to a subsystem where the data is manipulated in test environment.

**INDEXES**

The **INDEXES** option specifies how NGT Recover handles indexes when you specify the IMPORT command. (You can also use **INDEX** or **INDEXS** in place of **INDEXES**.)

**INDEXES AUTO**

**INDEXES AUTO**, the default value, specifies that NGT Recover recovers indexes if information is available for them in the migration data set. Any index not in the migration data set but defined on the target system is rebuilt.

**INDEXES RECOVER**

**INDEXES RECOVER** specifies that indexes on the target system are recovered if they have entries in the migration data set.

**INDEXES REBUILD**

**INDEXES REBUILD** specifies that all indexes on the target system are rebuilt regardless of whether they have entries in the migration data set.
INDEXES NO

INDEXES NO specifies that indexes on the target system are left unchanged.

RESET

The RESET keyword causes the log points in each data page to be reset to '0'. You must specify RESET when you are migrating data from one DB2 non-data-sharing system to another because the log point values on the first subsystem are meaningless or misleading on the target DB2.

NOCOPYPEND

If you specify the NOCOPYPEND option, NGT Recover resets COPY-pending status and issues message BMC96232 to inform you that COPY-pending status has been reset even though the space is not recoverable.

You may want to use the NOCOPYPEND option if you are migrating data from one DB2 subsystem to another in a query or testing system where you do not need to be able to run a recovery on the target space.

Because the input copy is not registered on the target system, the target space is not recoverable after the IMPORT. You cannot run a RECOVER to current on the target system because no usable copies are registered in SYSCOPY or BMCXCOPY (but you could run IMPORT again).

STATS

The STATS option allows you to carry forward existing RUNSTATS from a migration file. These RUNSTATS may then be optionally applied during an NGT Recover IMPORT. To import stats, you must have specified STATS YES in the NGT Copy EXPORT command.

STATS YES

STATS YES updates RUNSTATS information on the target system using information from the migration file. RTS stats are also updated.

STATS NO

RUNSTATS information is not updated. RTS stats are always updated. STATS NO is the default.

TABLESPACE specification

The TABLESPACE option specifies the target table space to be migrated.

The syntax diagram for the TABLESPACE specification is in Figure 49 on page 292.
TABLESPACE `databaseName.tableSpaceName`

If you do not specify the TABLESPACE option, all of the table spaces in the migration file will be imported. If you specify the TABLESPACE option, only the specified table spaces will be imported.

- `databaseName` is the name of the database containing the table space. The default is DSNDB04. The name cannot be DSNDB01, DSNDB06, DSNDB07, or any database that is defined with TYPE='W' in SYSIBM.SYSDATABASE.

- `tableSpaceName` is the name of the table space within the named database. BMCUTIL, BMCSYNC, and BMXCOPY (tables that are used by BMC utilities) cannot be used.

AUX

The AUX option specifies if auxiliary objects will be included with the migration of the base table spaces.

**Note**

If you specify the AUX keyword and the related table spaces are not included in the migration file, the migration will fail.

If you do not specify AUX on one of these commands, NGT Recover uses the value of the AUX installation option ("AUX=NO" on page 578), which defaults to NO.

AUX NO

NGT Recover does not include any auxiliary objects in the migration.

AUX ALL

NGT Recover includes XML and LOB objects in the migration.

AUX XML

NGT Recover includes all XML data objects along with the XML base table space. All implicitly created XML table spaces will be processed along with the base XML table space.

AUX LOB

NGT Recover includes all LOB data objects along with the LOB base table space. All table spaces created with the LOB attribute will be processed along with the base LOB table space.
HISTORY

The HISTORY option specifies if history tables should be included in the migration. You must use the AUX keyword to use the HISTORY keyword.

The meaning of the HISTORY keyword depends on which AUX option you specify, as follows:

- AUX NO HISTORY includes regular tables, system-time base tables, and their related HISTORY tables. It does not include related XML or LOB objects.
- AUX ALL HISTORY includes regular tables, system-time base tables and their related HISTORY tables, and all related XML or LOB objects.
- AUX XML HISTORY includes regular tables, system-time base tables and their related HISTORY tables, and all related XML objects.
- AUX LOB HISTORY includes regular tables, system-time base tables and their related HISTORY tables, and all related LOB objects.

Similar to DB2 RECOVER Version 10 VERIFYSET NO behavior, NGT Recover does not check that:

- The system-maintained temporal space and the history space are migrated as a set
- The system-maintained temporal space and the history space migrate to the same log point
- All DSNUMs are migrated in the same SYSIN

ARCHIVE

When you specify ARCHIVE, NGT Recover recovers archive-enabled base tables and their related archive tables.

You must specify the AUX option with ARCHIVE to tell NGT Recover whether to include auxiliary XML or LOB objects in the recovery:

- AUX NO ARCHIVE (or AUX ARCHIVE). It does not include related XML or LOB objects.
- AUX ALL ARCHIVE includes all related XML and LOB objects.
- AUX XML ARCHIVE includes all related XML objects.
- AUX LOB ARCHIVE includes all related LOB objects.
RICHK

RICHK specifies if NGT Recover will check for referential integrity constraints on the objects being recovered.

RICHK YES

RICHK YES checks for referential integrity constraints and sets check pending (CHKP) status for dependent objects not recovered in the same step to the same PIT.

RICHK NO

RICHK NO does not check for referential integrity. NO is the default value.

DEFINE

DEFINE specifies whether DEFINE NO objects will be instantiated.

Note

IBM DB2 Version 11 or later supports the DEFINE option.

DEFINE YES

NGT Recover instantiates DEFINE NO objects, if necessary, and updates the catalog and DBD to reflect that the object has been defined.

DEFINE NO

NGT Recover does not process DEFINE NO objects. NO is the default value.

OBJECTSET specification

The OBJECTSET option is available in IMPORT statement syntax to import groups that are defined using RECOVERY MANAGER. If you do not specify the TABLESPACE or OBJECTSET options, all of the table spaces in the migration file will be imported. If you specify the OBJECTSET option, only the specified table spaces will be imported.

The OBJECTSET specification is shown in “IMPORT syntax” on page 291.

OBJECTSET creator.name

The following rules apply to creator.name:
creator specifies the name of the creator of the group and can have up to 128 characters in length.

name specifies the group name and can have up to 128 characters in length.

You can delimit both creator and name with single or double quotation marks.

Both creator and name can contain the special characters $, #, and @ in any position.

NGT Recover uses dynamic grouping to determine the group contents at the time that you run the NGT Recover job containing the OBJECTSET option. Because NGT Recover reads the objects in the group each time the job is executed, objects may be added or removed from the group.

Note
NGT Recover does not read RECOVERY MANAGER recovery options for the group. If RECOVERY MANAGER recovery options change, you must run RECOVERY MANAGER to pick up the new options and generate the control cards.

Following are restrictions that apply when you use OBJECTSET in any of its forms on the RECOVER statement:

- You cannot specify DSNUM with OBJECTSET. NGT Recover uses the group definition for the object.

- NGT Recover allows only one OBJECTSET clause, and the statement cannot contain additional clauses.

When you use OBJECTSET, if NGT Recover tries to migrate a group of objects and more than 10 of the objects are unrecoverable, NGT Recover issues an error for each object that is unrecoverable and discontinues further processing of the group. This behavior is based on the ERRCONT option that has a default value of 10. To avoid this behavior, set the ON ERROR CONTINUE option with a value that will exceed the possible number of errors in the group, and the migration will complete.

If a table space is found in the OBJECTSET but it is not found in the migration file, the table space is considered unrecoverable. You can use the EXCLUDE specification to eliminate this object from the OBJECTSET.

EXCLUDE (dbname.ts or qualifier.ix)

Use the EXCLUDE option after an OBJECTSET specification to exclude one or more objects from migration. You can use wildcards or specific names to specify the exclusions. The following wildcards are valid:

- % and * match one or more characters in the object name.
- _ and ? match a single character in the object name.
List the excluded objects following the EXCLUDE keyword. Each item in the list must be in the form dbname.ts or qualifier.ix and you must separate the individual items with commas. Enclose the list in parentheses.

**INDEPENDENT OUTSPACE specification**

Use the INDEPENDENT (or INDEP) OUTSPACE option to redirect the output of the migration to a data set that is not used by DB2.

The syntax diagram for the INDEPENDENT OUTSPACE specification is in “IMPORT syntax” on page 291.

**INDEPENDENT OUTSPACE or INDEP OUTSPACE**

Use the INDEPENDENT (or INDEP) OUTSPACE option to redirect the output of the migration to a data set that is not used by DB2. INDEP OUTSPACE does not interfere with production processing. INDEP OUTSPACE enables you to do the following things:

You can specify the name of each independent data set or you can use a model name to represent the VSAM data set names of several target table spaces.

---

**Note**

NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

---

You can preallocate the INDEP OUTSPACE data sets using the following guidelines:

- For VCAT-defined spaces, ensure that the volumes or space allocated is large enough to hold the data set.

- If the target space is defined using a STOGROUP, and if primary and secondary quantities are not defined for the source space, NGT Recover uses a sliding-scale calculation for secondary extents, similar to the method DB2 uses.

For more information, see “Primary space allocation” on page 81 and “Secondary extents” on page 81.

You can allow NGT Recover to dynamically allocate the INDEP OUTSPACE data sets for STOGROUP-defined spaces. If the INDEP OUTSPACE data set does not exist, NGT Recover uses the attributes of the source space to dynamically allocate the target INDEP OUTSPACE data set. If the INDEP OUTSPACE data set exists, NGT Recover uses it as is (no DELETE or DEFINE executed).
NGT Recover includes support for multi-data-set spaces. For example, NGT Recover will dynamically allocate A002, A003, to A00n INDEP OUTSPACE data sets as needed.

To have NGT Recover dynamically allocate the INDEP OUTSPACE data set on a specific volume or volumes, pass a volume list by using the STOGROUP...USEORDER specification in the OPTIONS statement (see “STOGROUP specification description” on page 112). NGT Recover statements that use STOGROUP... USEORDER are shown in the following example:

```
OPTIONS STOGROUP TESTSTO USEORDER(NEW004,NEW005,NEW006)
```

The default model name is identical to the DB2 name of the object being recovered, but with BMCDBC substituted for DSNDBC in the CLUSTER portion of the VSAM data set:

```
vcat.BMCDBC.databaseName.objectName.p0001.znnn
```

The variables are defined as follows:

- **vcat** is the ICF catalog name.
- **databaseName.objectName** is the name of the database and object containing the data set being recovered.
- **p** corresponds to the value that is used by DB2 for this node.
- **znnn** is the data set number.

`z` represents the letter A, B, C, D, or E. These letters correspond to the first digit—0, 1, 2, 3, or 4—of the partition number. `nnn` represents the rest of the partition number. `znnn` represents the partitions numbers as follows:

- A001 through A999 for partitions 1 through 999
- B000 through B999 for partitions 1000 through 1999
- C000 through C999 for partitions 2000 through 2999
- D000 through D999 for partitions 3000 through 3999
- E000 through E096 for partitions 4000 through 4096

When you want to specify a name of your own choosing, use the MODEL option to specify the name of the CLUSTER portion of the VSAM data set that you want to use. The data set name that you specify may include any of the symbolic variables that are listed in the description of the MODEL keyword in the following section.

When you use INDEP OUTSPACE, you must specify it after the object specification.
MODEL

The MODEL option specifies the non-DB2 data set cluster name to which you want to redirect the output from the recovery of the specified object. The data set name that you specify may optionally include any of the symbolic variables in “Using symbolic variables” on page 494.

OUTCOPY specification

This topic contains option descriptions for the OUTCOPYDDN and RECOVERYDDN specifications which are included in the OUTCOPY specification.

The syntax diagram for the OUTCOPY specification is in “IMPORT syntax” on page 291.

OUTCOPY

You can specify OUTCOPY to make copies of the object or object data set being imported. For each object or data set that you process, you can request up to four image copies. The default is no copies. If you are migrating to the real DB2 data sets and you request image copies, you can choose which copies, if any, to register in the SYSIBM.SYSCOPY table. However, if you are using the INDEPENDENT OUTSPACE option to import to non-DB2 data sets and you request image copies, you cannot register those copies.

Output copies are produced in the SYSTEMPAGES YES format when possible.

If the output copies that you want to make are allocated in the JCL, the OUTCOPYDDN and RECOVERYDDN options specify the DD names for those data sets. If the output copies are dynamically allocated, the OUTCOPYDDN and RECOVERYDDN options refer to the appropriate OUTPUT descriptor names. For more information, see “OUTCOPYDDN specification” on page 198 and “RECOVERYDDN specification” on page 199. The copies that you want to register are specified by the REGISTER option (“OUTCOPY specification” on page 193).

When you create an OUTCOPY and want to force the copy to the cylinder-managed space of an extended address volume (EAV), specify the SPACE keyword (“OUTCOPY specification” on page 193).

Note

You can use NGT Recover to automatically size the OUTCOPY files generated by NGT Recover. For more information, see “Automatic sizing of output copies dynamically allocated on DASD” on page 499.

How the copies are made depends on whether the object is partitioned or nonpartitioned.
If the object is partitioned, and OUTCOPY BYPART is specified or a specific DSNUM is coded in the object specification, a separate output copy is made for each partition. If you specified OUTCOPY ASCODED and DSNUM ALL with the RECOVER command, a single output copy is made for the object as a whole.

For more information about using OPTIONS OUTCOPY BYPART, see “OUTCOPY” on page 116.

If the object is nonpartitioned and you are importing multiple data sets by specifying DSNUM ALL, each output copy is for all of the data sets. If you are importing multiple data sets by specifying each one individually with DSNUM integer, a separate output copy is made for each data set.

OUTCOPY NO

OUTCOPY NO is the default and specifies that no output copies are required.

OUTCOPY YES

OUTCOPY YES specifies that you want image copies or DSN1COPY-type copies of the table space being imported. You can make up to four image copies or up to four DSN1COPY-type copies. A DD statement or an OUTPUT command statement must exist for each copy. The ddnames for the copy DD statements are specified by the OUTCOPYDDN and RECOVERYDDN options.

Note
You cannot use the OUTCOPY YES option to produce Instant Snapshot copies.

OUTCOPYDDN specification

The syntax diagram for the OUTCOPYDDN specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

RECOVERYDDN specification

The syntax diagram for the RECOVERYDDN specification is in “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax” on page 151.

CLONE

The CLONE option indicates that NGT Recover is to import clone table data. When you import clone data, the following limitations apply:
You cannot refer to the same object with the CLONE option and without the CLONE option in the same recovery step; you cannot process the base and its clone in the same command.

Related objects (table space and all indexes for its table, all related LOB objects) should use the same CLONE specification in the same recovery step.

**ANALYZE**

The ANALYZE option prints an NGT Recover plan before executing that plan.

The following information about objects is included in the plan:

- Names of any image copy data sets on which activities are or will be based
- Names of any log data sets on which activities are or will be based
- Name of any change accumulation files on which activities are or will be based
- Log ranges, if any, on which activities are or will be based
- Number of log pages to be read
- Record sizes for index sort work data sets
- Phases to occur during execution
- Steps to occur within each phase

Use this information to allocate space more accurately for log sort work data sets, index sort work data sets, and index key work data sets, thereby limiting abends that are caused by inadequate data set allocations. You can use these statistics with historical information about the times required to perform the various operations to estimate recovery time.

**Note**

If you coded ANALYZE with the OPTIONS command, you must code the same value here. If you code multiple NGT Recover command statements you must use the same value for the ANALYZE option in all of the command statements. If you specify a value for ANALYZE on one command statement and take the default on the others, the value that you specified on the single command statement becomes the default for the others.

**ANALYZE YES**

ANALYZE YES is the default and provides the information listed in the preceding section.
ANALYZE NO

If you specify ANALYZE NO, NGT Recover does not provide any plan information, but does provide object summaries.

ANALYZE ONLY

If you specify ANALYZE ONLY, NGT Recover provides the same information as it provides with ANALYZE YES and then stops the job when the ANALYZE phase is complete. You can use the data from the ANALYZE phase to determine the resources required and what will happen during the run. You cannot restart an NGT Recover job that specifies ANALYZE ONLY, but you can start a new job.

Use the ANALYZE ONLY option to determine which phases will occur with a specific request, or which copy data sets and log data sets will be used.

If you provide a SYSPICK DD statement, NGT Recover generates a list of all of the input tape and cartridge volumes that are allocated. For more information about SYSPICK, see “NGT Recover data sets and NGT Recover DD statements” on page 319.

If you specify OPTIONS EARLYRECALL with ANALYZE ONLY, the recall of the data sets needed for execution will be initiated. You must code OPTIONS EARLYRECALL explicitly to initiate the recalls, because OPTIONS NOEARLYRECALL is the default for ANALYZE ONLY executions.

REDEFINE

Use the REDEFINE option to avoid the normal deletion and reallocation of the object data sets for STOGROUP-defined objects. This option is ignored for VCAT-defined object data sets. The value that you specify for REDEFINE applies to all STOGROUP-defined objects included with a single NGT Recover command. To recover multiple STOGROUP-defined objects with some reallocated and others not reallocated, you must use one NGT Recover command statement for the objects to be reallocated and a second NGT Recover command statement for those objects not to be reallocated.

Regardless of the value of REDEFINE, if the data sets for a STOGROUP-defined object do not exist, NGT Recover creates the VSAM data sets.

Note

When a point-in-time recovery of a multi-data-set, nonpartitioned object is performed, more or fewer data sets may result than were previously used. If data sets were added after the point-in-time recovery point, there are fewer data sets after the point-in-time recovery. If the object was reorganized after the point-in-time recovery point, there may be more data sets.
REDEFINE YES

REDEFINE YES is the default. If you specify REDEFINE YES, NGT Recover deletes and reallocates the VSAM data sets for STOGROUP-defined objects prior to the recovery. For more information, see Storage group-defined data sets on page 80.

NOSCRATCH

If you specify NOSCRATCH with REDEFINE YES, NGT Recover invokes IDCAMS to uncatalog without scratching. You can use NOSCRATCH to support disaster recovery scenarios where the following conditions are true:

- DASD volumes at the recovery site differ from those at the primary site.
- The operating system catalog and the DB2 catalog and directory are restored before running NGT Recover.
- DB2 STOGROUPS are changed to point to the recovery site packs.

Using this option eliminates mount messages for the primary site packs that do not exist at the recovery site.

Note

REDEFINE YES has no effect if you specify INDEP OUTSPACE. For the instructions for creating data sets for INDEP OUTSPACE, see “INDEPENDENT OUTSPACE specification” on page 185. If an Instant Snapshot copy is used to recover the object, REDEFINE YES has no effect and the data sets are processed as if REDEFINE NO had been specified.

REDEFINE NO

REDEFINE NO tells NGT Recover not to delete and reallocate the VSAM data sets for the object. This option enables you to treat STOGROUP-defined objects as VCAT-defined objects and to reallocate the VSAM data sets manually, or use the data sets already assigned to the object. New volumes are used as needed from the STOGROUP, even with REDEFINE NO specified.

Note

With REDEFINE NO specified, when recovery of a multi-data-set, nonpartitioned object is performed as DSNUM ALL, any necessary additional data sets are created and unused data sets are deleted.

When you specify REDEFINE NO, NGT Recover does not free unused extents in a retained data set. If you want the recovery to free unused extents, specify REDEFINE YES, which is the default.
**REUSE**

The REUSE option specifies that the space will not be deleted or redefined. This option cannot be used with the REDEFINE option.

**UPDATE VERSIONS**

Use the UPDATE VERSIONS option to have NGT Recover call the DB2 REPAIR VERSIONS utility after a table space or index space recovery of a versioned table space is complete. The DB2 REPAIR VERSIONS utility updates the version information in the DB2 catalog and directory so that the source and target information match.

If you use NGT Recover and this option to perform migration, you need to:

- Make sure that your copies contain system pages or directory pages for all versions of the space
- Import table spaces in a separate step from rebuilding indexes
  You can recover indexes and tables spaces in the same step, but rebuilds must be in a separate step.
Building and running NGT Recover jobs

This chapter explains how to build a job and describes all of the utility parameters. Instructions about how to perform the following tasks are also included:

- Starting an NGT Recover job
- Restarting a job
- Terminating a job

Building NGT Recover jobs

Building a job for the NGT Recover product involves creating a set of JCL that includes:

- A JOB statement ("JOB statement" on page 312)
- An EXEC statement with the appropriate utility parameters ("EXEC statement" on page 312 and "Utility parameters on the EXEC statement" on page 313)
- STEPLIB or JOBLIB DD statements as needed ("STEPLIB DD or JOBLIB statements" on page 319)
- DD statements as needed for the data sets needed for the recovery ("NGT Recover data sets and NGT Recover DD statements" on page 319 to "Determining unloaded keys file space requirements" on page 333)
- NGT Recover statements using the appropriate command syntax ("Specifying NGT Recover syntax" on page 334)

A simple example of an NGT Recover job is illustrated in the following figure. For more examples of NGT Recover jobs, see “Examples of NGT Recover jobs” on page 367.

Figure 54: JCL for a simple NGT Recover execution

```
//DMBRECV2 JOB (5212), 'SIMPLE2', CLASS=A,
// MSGCLASS=X, NOTIFY=AFR
```
JOB statement

Include an NGT Recover JOB statement that conforms to your site’s standards. You can include the REGION parameter on either your JOB statement or your EXEC statement.

For recommendations, see “REGION parameter” on page 312.

EXEC statement

The NGT Recover EXEC statement specifies PGM = AFRMAIN.

The EXEC statement also specifies NGT Recover utility parameters, which are described in “Utility parameters on the EXEC statement” on page 313.

You can include the REGION parameter on either your EXEC statement or your JOB statement. See “REGION parameter” on page 312 for recommendations.

REGION parameter

Include the REGION parameter on either your JOB statement or your EXEC statement to specify the region size (the amount of virtual storage used by the utility). For the best performance, BMC recommends that you specify REGION=0M to allocate all available virtual storage to the NGT Recover job. If your data center does not permit you to specify REGION=0M, specify 8 MB to ensure adequate storage. A typical NGT Recover step requires between 5 MB and 8 MB of virtual storage for code, control blocks, and I/O buffers.

Note

If you specify a value for REGION other than 0M, ensure that you have an appropriate value set for the MEMLIMIT parameter, either as your site’s default SMF option or on your JOB statement or EXEC statement. BMC recommends that you have a MEMLIMIT value of at least 2 GB. For more information, see “Setting the MEMLIMIT parameter” on page 66.
For a discussion of the virtual memory usage when many log files are read concurrently, see “Does LOGSORT have other advantages or disadvantages?” on page 460.

Utility parameters on the EXEC statement

The NGT Recover EXEC statement includes the following utility parameters:

- DB2 subsystem ID or group attach name for data sharing (ssid)
- Utility ID (utilID)
- Restart parameter (restartParm)
- Message level parameter (msgLevel)
- Checkpoint override parameter (checkPt)
- Parameter for restoring the initial status of DB2 objects (rdb2Stat)
- Installation options module for NGT Recover (afrOpts)

The following illustration shows the format of the EXEC statement:

```
//stepName EXEC PGM=AFRMAIN,REGION=0M,
//         PARM='ssid,utilID,restartParm,msgLevel,checkPt,rdb2Stat,afrOpts'
```

The parameters ssid, utilID, and restartParm are positional and must be specified in the order shown.

You must specify the DB2 subsystem ID or group attach name for DB2 data sharing. However, if you do not specify the utility ID or the restart parameter and want to use the defaults for these parameters, you must include a comma as a place holder for each parameter. For example, in a statement with the following format, the subsystem ID or group attach name (ssid) and the restart parameter (restartParm) are specified and the default is used for the utility ID:

```
//         PARM='ssid,,restartParm'
```

After you specify the DB2 subsystem or group attach name, the utility ID, and the restart parameter, you can specify the remaining parameters in any order; they are non-positional. If you do not specify one of the non-positional parameters, NGT Recover uses the default value for the utility parameter.

DB2 subsystem identifier (ssid)

The DB2 subsystem identifier is the one-character to four-character DB2 subsystem ID that identifies where the objects to be recovered reside. If the DB2 subsystem identifier is not coded in the runtime parameters on the JCL EXEC statement for the utility, the default value from the DSNHDECP module that is in the STEPLIB concatenation will be used. (The DSNHDECP module was created when DB2 was
installed. The installation-generated DSNHDECP module typically resides in the SDSNEXIT library.

You can also use the group attachment name (for DB2 data sharing) in place of the ssid parameter. Using the group attachment name, you can use the same JCL but run on any member of a data sharing group. When a job is submitted with a group name, a subsystem in the group must be executing on the system where the job is submitted.

**Utility Identifier (utilID)**

This parameter specifies the ID that uniquely names a utility execution or job step. If you do not specify this parameter, NGT Recover uses the default, userID.jobName.

The rules for utility ID are as follows:

- The utility ID can be 1 to 16 characters.
- The utility ID consists of alphanumeric characters, plus the following characters: #, $, @, !, ¬, ., and €.

When you run multiple NGT Recover jobs concurrently, each job must use a unique utility ID.

**Restart parameter (restartParm)**

The restart parameter can have one of the values described in this topic.

The value that you choose for this parameter determines which of the following execution types is invoked:

- A new NGT Recover execution
- An NGT Recover execution that you want to restart
- A run to print control section information to track maintenance that has been applied without performing any other processing

For more information about resubmitting a failed NGT Recover job, see “Restarting an NGT Recover job” on page 335.

**Blank or not specified**

This value is the default. If the utility ID is not present in the BMCUTIL table, this value starts a new NGT Recover job. If the utility ID is already present in the BMCUTIL table, NGT Recover terminates with an error.
MAINT

MAINT limits processing to listing the installation options, fixes applied to NGT Recover, and the names of BMC tables. No recovery processing is performed.

--- Note ---

Even if the connection to DB2 fails, NGT Recover still prints the maintenance information.

NEW

NEW starts a new NGT Recover utility execution. You can reuse a utility ID that already exists in the BMCUTIL and BMCSYNC tables. (These tables are described in “Common utility tables” on page 605.) When you use NEW, any existing utility job with the same ID is replaced, and a new utility job is started. NGT Recover will not allow the job to start if another job with the same utility ID on the same DB2 subsystem or data sharing group is active.

NEW/RESTART

If the utility ID already exists in the BMCUTIL table, NEW/RESTART restarts the utility job from the last recorded sync point or (if the last checkpoint taken was not for a sync point) from the beginning of the last incomplete phase. If the utility ID does not already exist, NEW/RESTART initiates a new NGT Recover utility job. This option allows a job to be submitted again without changing the JCL in most cases. Some cases may require a change to DD statements with a disposition of NEW. For more information about sync points and checkpoints, see “Restarting an NGT Recover job” on page 335.

NEW/RESTART(PHASE)

NEW/RESTART(PHASE) restarts the utility at the beginning of the last incomplete NGT Recover phase if the utility ID already exists in the BMCUTIL table. Otherwise, NEW/RESTART(PHASE) starts the job as a new NGT Recover utility job. This option allows a job to be submitted again without changing the JCL in most cases. Some cases may require a change to DD statements with a disposition of NEW.

RESTART

If the utility ID already exists, RESTART restarts the utility from the last sync point or (if the last checkpoint taken was not for a sync point) from the beginning of the last incomplete phase. If the utility ID does not already exist in the BMCUTIL table, NGT Recover terminates and issues an error message. For more information about sync points, see “Restarting an NGT Recover job” on page 335.
**RESTART(PHASE)**

RESTART(PHASE) restarts the utility at the beginning of the last incomplete phase. A row for the utility ID must exist in the BMCUTIL table or the job terminates with an error message.

**TERM**

Specifying TERM terminates a stopped or failed utility by removing all sync point and restart information for the utility ID from the BMCUTIL and BMCSYNC tables. You can use the TERM parameter to "clean up" after an error condition; however, you may prefer to restart the recovery. NGT Recover does not issue an error message when TERM is specified, and no row exists for the utility ID in the BMCUTIL table.

---

**Note**

For additional instructions about "clean up" after use of RECOVER UNLOADKEYS, see “Cleaning up RECOVER UNLOADKEYS entries” on page 619.

---

**Message level parameter (msgLevel)**

This utility parameter determines which output files and messages NGT Recover returns.

Valid values for MSGLEVEL are 0, 1, and 2. If you do not specify msgLevel, NGT Recover uses the default, MSGLEVEL(1). For a description of the output files that are returned by different values of MSGLEVEL, see Table 9 on page 320. For examples of the different output files and their content, see “Using MSGLEVEL to control NGT Recover output” on page 341.

**Checkpoint override parameter (checkPt)**

This utility parameter overrides the CHECKPT installation option.

CHECKPT provides a means of controlling the overhead associated with taking checkpoints. Taking a checkpoint refers to the process of recording the end of a phase in the BMCUTIL and BMCSYNC tables. When a checkpoint is taken, message BMC40091 is issued. If you do not specify the checkpoint override parameter, the default is the value of the CHECKPT installation option.

For more information about CHECKPT, see “Checkpoints for NGT Recover restart” on page 78 and “NGT Recover installation options” on page 573.

The checkpoint override parameter can have a value of CHECKPT(NO) or CHECKPT(PHASE).
CHECKPT(NO)

CHECKPT(NO) causes no checkpoints to be taken, except those necessary to synchronize NGT Recover execution with the execution of other BMC utilities and MERGE checkpoints that are necessary to guarantee the integrity of output copy, point-in-time recovery, or index rebuild registrations. CHECKPT(NO) is recommended for short NGT Recover jobs in which you do not want to incur checkpoint overhead or that you do not mind rerunning entirely if necessary.

CHECKPT(PHASE)

CHECKPT(PHASE) causes a checkpoint to be taken at the end of each processing phase if more than the number of minutes specified by the value of the CHECKINT installation option have passed since the last checkpoint was taken. For information about the CHECKINT installation option, see “NGT Recover installation options” on page 573.

Specify CHECKPT(PHASE) for longer running jobs when it would be costly to rerun the entire job.

Note

Specifying registered output copies in a RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX command causes a checkpoint to be taken at the end of the MERGE phase for the table space, index space, or index, even if the required number of minutes has not elapsed. This action protects the update to SYSIBM.SYSCOPY, which registers the copy. No sync points are taken. Checkpoints may be forced to guarantee registration of a point-in-time recovery or a rebuild of an index.

RDB2STAT override parameter (rdb2Stat)

This utility parameter overrides the RDB2STAT installation option.

RDB2STAT indicates the status to which DB2 objects should be returned at the successful completion of an NGT Recover job. If you do not specify the RDB2STAT override parameter, NGT Recover defaults to the setting of the RDB2STAT installation option. For information about the RDB2STAT installation option, see “NGT Recover installation options” on page 573.

The RDB2STAT override parameter can have a value of RDB2STAT(YES), RDB2STAT(NO), RDB2STAT(RW), or RDB2STAT(RO).

RDB2STAT(YES)

RDB2STAT(YES) causes NGT Recover to issue DB2 commands to restore the DB2 status of each space after NGT Recover completes the requested
operations. For examples of how NGT Recover restores the initial status of a space, see “Restoring initial status” on page 74.

RDB2STAT(NO)

RDB2STAT(NO) causes NGT Recover not to issue DB2 commands to restore the DB2 status of spaces after an NGT Recover run is complete. The spaces remain stopped.

RDB2STAT(RW)

RDB2STAT(RW) causes NGT Recover to issue DB2 commands to set the status of each space to read-write (RW) mode (regardless of the initial status of the space) after the requested operations have completed successfully.

Note

If RECOVER UNLOADKEYS was specified for an index, the associated table space is placed in RO status.

RDB2STAT(RO)

RDB2STAT(RO) causes NGT Recover to issue DB2 commands to set the status of each space to read-only (RO) mode (regardless of the initial status of the space) after the requested operations have completed successfully.

Installation options module parameter for NGT Recover (afrOpts)

The installation options module parameter overrides the installation options module, AFR$OPTS, for NGT Recover. Using this feature, you can provide a different options module at runtime. The length of the parameter determines how the parameter is used:

- If this parameter is greater than four characters in length, the options module specified by the parameter is loaded.

- If this parameter is less than or equal to four characters in length, the parameter is considered a suffix override for the OPTS part of the installation options module name. For example, specifying AFROPTS(TEMP) results in an attempt to load AFR$TEMP as the NGT Recover options module.

As an alternative to afrOpts, you can also specify opts on the EXEC statement. For example, either of the following examples is valid:

```
PARM='DFC2,F312396B,NEW,MSGLEVEL(1),AFROPTS(AFR$DB2A)'
PARM='DFC2,F312396B,NEW,MSGLEVEL(1),OPTS(AFR$DB2A)'
```

For information about the AFR$OPTS installation options module, see “NGT Recover installation options” on page 573.
STEPLIB DD or JOBLIB statements

The NGT Recover STEPLIB DD statement must specify the following libraries, unless they are included in your system’s LINKLIST or in a JOBLIB statement:

- Load libraries that contain the files (including the installation options module) for the following BMC products and components:
  - NGT Recover
  - BMCSORT (AUP)
  - DB2 Solution Common Code (SCC)

- Libraries that contain any EDITPROCS, VALIDPROCS, FIELDPROCS, and user-written routines
  EDITPROCs are required only when a REBUILD INDEX or RECOVER UNLOADKEYS command is requested for an index on a table with an EDITPROC specified.

- DB2 load library

All load libraries in the STEPLIB or JOBLIB concatenation must be APF authorized.
(For more information, see “APF authority” on page 70.)

NGT Recover data sets and NGT Recover DD statements

Your recovery needs and the NGT Recover strategy that you use determine the data sets needed for recovery. Some data sets are required; others are optional. Some data sets require DD statements in the JCL; others can be dynamically allocated by NGT Recover.

DD statements common to all NGT Recover executions

You can control the amount of output that you receive by specifying a value for the MSGLEVEL utility parameter. Valid values for MSGLEVEL are 0, 1, and 2. If you do not specify MSGLEVEL, NGT Recover uses the default, MSGLEVEL(1).

For more information, see “Using MSGLEVEL to control NGT Recover output” on page 341.

The following table provides a list of NGT Recover data sets along with specific information about each data set, including

- Description of the data set
- Whether the data set is required or optional
- Whether a specific MSGLEVEL is required to produce the data set
Whether the data set can be dynamically allocated or whether it requires a DD statement

### Table 9: NGT Recover data sets

<table>
<thead>
<tr>
<th>Data set name</th>
<th>Required or optional</th>
<th>Dynamically allocated (yes or no)</th>
<th>MSGLEVEL value required #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Required</td>
<td>No</td>
<td>Not applicable</td>
<td>Defines the input data set that contains the NGT Recover commands and their options. This data set must be fixed blocked (RECFM=FB) with a record length of 80 (LRECL=80). Columns 73-80 are reserved for sequence numbers; any characters in these columns are ignored by NGT Recover. For details of the NGT Recover commands and options that you can specify, see “NGT Recover syntax” on page 83. Unicode is not supported in the SYSIN file.</td>
</tr>
<tr>
<td>AFRPRINT</td>
<td>Required</td>
<td>Yes</td>
<td>1 or 2</td>
<td>Contains a list of the phases executed for each object and the name of the AFRPRnnn data set where NGT Recover put the output for the phase</td>
</tr>
<tr>
<td>AFRPRnnn</td>
<td>Required</td>
<td>Yes</td>
<td>0, 1, or 2</td>
<td>Provides messages for each execution phase. NGT Recover allows up to (MAXKSORT + MAXLSORT + 1) AFRPRnnn files. NGT Recover names the files AFRPR001, AFRPR002, AFRPR003, and so on.</td>
</tr>
<tr>
<td>AFRSUMRY</td>
<td>Required</td>
<td>Yes</td>
<td>0, 1, or 2</td>
<td>Lists maintenance applied, phases completed, utility return codes</td>
</tr>
<tr>
<td>AFRSTMT</td>
<td>Required</td>
<td>Yes</td>
<td>0, 1, or 2</td>
<td>Lists input statements, commands, and options as specified in SYSIN, installation option values, log file resources, and messages generated by analyze, planning, and termination stages</td>
</tr>
<tr>
<td>AFROSUM</td>
<td>Optional, based on MSGLEVEL</td>
<td>Yes</td>
<td>1 or 2</td>
<td>Lists all of the objects being recovered or rebuilt and their status, options, and resources</td>
</tr>
<tr>
<td>Data set name</td>
<td>Required or optional</td>
<td>Dynamically allocated (yes or no)</td>
<td>MSGLEVEL value required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------</td>
<td>-----------------------------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>AFRPLAN</td>
<td>optional, based on MSGLEVEL</td>
<td>Yes</td>
<td>2</td>
<td>Records the execution plan When you use a BMC Recovery Management for DB2 solution password, AFRPLAN includes values for recovery estimation.</td>
</tr>
<tr>
<td>AFRERR</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Records warnings or error messages</td>
</tr>
<tr>
<td>AFRTIME</td>
<td>Optional</td>
<td>Yes, if needed</td>
<td>Not applicable</td>
<td>Available only with the BMC Recovery Management for DB2 solution For the recovery estimation feature, reports the ten table spaces for which the longest elapsed time was recorded to recover each table space and all of its indexes</td>
</tr>
<tr>
<td>AFRTRACE</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Provides trace information when errors occur during execution</td>
</tr>
<tr>
<td>SYSERR or AFRDUMP</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Captures snap dumps issued during error processing If SYSERR is not coded in the JCL, NGT Recover looks for AFRDUMP. If DD names are not coded for either SYSERR or AFRDUMP, NGT Recover attempts to dynamically allocate SYSERR. BMC does not recommend the suppression of dump output because it is used as a troubleshooting aid.</td>
</tr>
<tr>
<td>SYSOU nnn</td>
<td>Required</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Used for BMCSORT message output</td>
</tr>
<tr>
<td>SYSUDUMP</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Captures system abend dumps If SYSUDUMP is not in the JCL, it is dynamically allocated to SYSOUT. BMC recommends the use of SYSUDUMP as a troubleshooting aid.</td>
</tr>
<tr>
<td>SYSSCAN</td>
<td>Optional</td>
<td>No</td>
<td>Not applicable</td>
<td>Contains the output from the LOGSCAN command If the SYSSCAN DD is not specified, the LOGSCAN output is in AFRPRINT.</td>
</tr>
<tr>
<td>Data set name</td>
<td>Required or optional</td>
<td>Dynamically allocated (yes or no)</td>
<td>MSGLEVEL value required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------</td>
<td>-----------------------------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>AFRGDG</td>
<td>Optional</td>
<td>No</td>
<td>Not applicable</td>
<td>Used with dynamic allocation to provide a GDG base if one does not exist. This data set contains the control cards needed to perform an IDCAMS DEFINE operation and the symbolic variable &amp;BASE. For more information about the use of GDGs, see “GDGs and symbolic variables in data set name construction” on page 493.</td>
</tr>
<tr>
<td>SYSPICK</td>
<td>Optional</td>
<td>No</td>
<td>Not applicable</td>
<td>Lists all of the input tape and cartridge volumes that are allocated during recovery, including those implicated because of a need to recall archived data sets to disk prior to recovery. SYSPICK output consists of two reports. Both reports contain volume serial numbers, device types, data set names, and sequence numbers. One report orders the data by volume serial number and the other report orders data based on the order in which NGT Recover uses the data sets. Allocation of the SYSPICK data set DCB should be RECFM=VB,LRECL=137.</td>
</tr>
<tr>
<td>AFRDBG</td>
<td>Optional</td>
<td>No</td>
<td>Not applicable</td>
<td>Used to help with performance and problem diagnosis associated with Instant Snapshots, if you have EXTENDED BUFFER MANAGER (XBM) version 5.6 or later and you use Instant Snapshots for recovery.</td>
</tr>
</tbody>
</table>
Data set name | Required or optional | Dynamically allocated (yes or no) | MSGLEVEL value required | Description
---|---|---|---|---
a | The values shown are the required values of MSGLEVEL to produce the output file. The default value of MSGLEVEL is 1.
b | When "not applicable" is shown, NGT Recover produces the data set regardless of the value of MSGLEVEL. In some cases the files are dynamically allocated and in other cases they are not.
c | For descriptions of the NGT Recover commands and options, see NGT Recover syntax on page 83. Sections are included with information on the use of Long Names and Unicode.
d | However, NGT Recover can process spaces that contain tables, indexes, or storage groups with Unicode names. (Spaces cannot contain Unicode names). NGT Recover commands that use wild cards do not include objects that match the pattern but contain Unicode characters that are not translatable to EBCDIC in the wild card position. This is because SYSIN is EBCDIC and wild card processing is done in EBCDIC.
e | NGT Recover produces these data sets regardless of the value of MSGLEVEL when you specify OPTIONS ANALYZE ONLY.

Data sets and DD statements for RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX

The options specified, or for which defaults are accepted, in a RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX command statement may require that you specify DD statements in the JCL in addition to those common to all NGT Recover executions.

See “NGT Recover data sets and NGT Recover DD statements” on page 319.

Use DD statements to specify the following data sets:

- Any requested output image copy data sets that are not dynamically allocated
- Log sort work data sets

For examples that use dynamic allocation and DD statement construction for output copy DD statements, see “Examples of NGT Recover jobs” on page 367.

Input image copy data sets

Normally dynamically allocated, input image copy data sets are needed for recovery unless recovery is possible by using only the log (for example, with LOGAPPLY ONLY or through a point-in-time recovery with BACKOUT).

DD statements are allowed, but BMC does not recommend their use for input image copy data sets. Use a DD statement for an input copy only to override a dynamic allocation. If you use a DD statement, it must contain a unit and a VOL=SER for
uncataloged data sets. NGT Recover matches the information in the DD statement to
the data set that NGT Recover would otherwise dynamically allocate by data set
name and VOL=SER if uncataloged. Any DDNAME can be specified.

You do not need to code DD statements to control tape mounts for stacked tapes.
NGT Recover automatically selects the proper order and handling to accommodate
this.

---

**Note**

Tape stacks with more than 255 volumes are not supported. You must split the
recovery into two or more steps when using such a tape stack.

---

### Output image copy data sets

Output image copy data sets are required only when you are requesting one or more
output copies. Either ddnames or ddname prefixes are given in the syntax (or
defaults are used), and these names or prefixes correspond to the JCL. Another
option is to use dynamic allocation of output image copy data sets (see “Allocating
output image copy data sets dynamically” on page 491).

---

**Note**

Dynamic allocation of output copies is required for spaces with more than 254
partitions.

---

The ddnames are constructed differently for partitioned and nonpartitioned spaces:

- **DDname Construction for Partitioned Spaces**
  
  If you have coded OPTIONS OUTCOPY BYPART, or if the OUTCOPY installation
  option is set to BYPART, and you have coded DSNUM on the RECOVER
  TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX command
  statement, you specify the ddname as follows:

  \[
  DDName_{nn} \text{ or } DDName_{nnn}
  \]

  \[\text{DDName is the prefix that you specified, and } nn \text{ (for spaces with fewer than 100 partitions) or } nnn \text{ (for spaces with 100 or more partitions) is the partition number. For spaces with fewer than 100 partitions, } DDName \text{ is limited six characters. For spaces with 100 or more partitions, } DDName \text{ is limited to five characters.}\]

  The following table lists the default ddname *prefixes*.

<table>
<thead>
<tr>
<th>Type of copy</th>
<th>Less than 100 partitions</th>
<th>Equal to or greater than 100 partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local primary copy</td>
<td>BMCCPY</td>
<td>BMCCY</td>
</tr>
</tbody>
</table>

---

324   *BMC Next Generation Technology Recover for DB2 for z/OS Reference Manual*
<table>
<thead>
<tr>
<th>Type of copy</th>
<th>Less than 100 partitions</th>
<th>Equal to or greater than 100 partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local backup copy</td>
<td>BMCCPZ</td>
<td>BMCCCZ</td>
</tr>
<tr>
<td>Remote primary copy</td>
<td>BMCRCY</td>
<td>BMCRY</td>
</tr>
<tr>
<td>Remote backup copy</td>
<td>BMCRCZ</td>
<td>BMCRZ</td>
</tr>
</tbody>
</table>

The copies are made for the space as a whole if OPTIONS OUTCOPY ASCODED is coded, or if the OUTCOPY installation option is defaulted to ASCODED, and you have not coded DSNUM on the RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX command statement.

For these copies, you specify a ddname of up to eight characters. The default ddnames are the same as those shown in the preceding table in the Less Than 100 Partitions column.

**DDname Construction for Nonpartitioned Spaces**

When you are recovering a nonpartitioned space, the construction of the ddname depends on how the recovery is requested. If the request is made for DSNUM ALL (the default), the ddname is the ddname for the copy data set, up to eight characters. The default ddnames are the same as those shown in the preceding table in the Less Than 100 Partitions column.

If the request is to recover a specific data set of the space, for example DSNUM 2, the ddname becomes a prefix.

The default ddname prefixes depend on the highest DSNUM that you specified in the input commands. If DSNUM is less than 100, the default ddname prefixes are the same as those shown in the preceding table in the Less Than 100 partitions column.

If the highest DSNUM specified is greater than or equal to 100, a maximum of five characters may be specified for the prefix, and the default values are the same as those shown in the preceding table in the Equal to or greater than 100 Partitions column.

**Log data sets**

Log data sets are used when RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX command processing involves applying log records. The data sets are dynamically allocated by NGT Recover. No DD statements are allowed for log data sets. To control whether the utility uses archive or active log data sets and which copy is used, refer to OPTIONS RESOURCE SELECTION (see “RESOURCE SELECTION” on page 121).

**Log sort work data sets**

Log sort work data sets are used when log records are to be applied. For more information, see “Defining sort work data sets” on page 329.
DD statements for REBUILD INDEX jobs

The options specified, or for which defaults are accepted, with the REBUILD INDEX command may require that you specify DD statements in the JCL in addition to those common to all NGT Recover executions.

See “DD statements common to all NGT Recover executions” on page 319.

Note
Because REBUILD INDEX and RECOVER INDEX are synonyms when you specify INDEXLOG=NO, the information in this section applies to RECOVER INDEX INDEXLOG=NO jobs. It could also apply to RECOVER INDEX INDEXLOG=AUTO jobs if no index image copies are found. (For more information about INDEXLOG, which is an installation option and which can also be set on the OPTIONS statement, see “INDEXLOG=NO” on page 584 and “INDEXLOG” on page 124.) SIMRBLD and SIMRCVR may also require these DD statements.

Use DD statements to specify the following types of data sets:

- Sort work data sets for keys
  - Parallel sorts
  - Non-parallel sorts
- Sort message data sets for parallel sorts
- Work data sets for extracted keys

If RECOVER UNLOADKEYS command statements exist for the same space in the recovery, see “DD statements for RECOVER UNLOADKEYS jobs” on page 327.

Data sets needed for parallel index sorts and rebuilds

When you use the MAXKSORT option to specify that multiple key sorts (and index rebuilds) are to run in parallel subtasks, the following data sets are dynamically allocated by NGT Recover, unless you provide DD statements for them:

- SxxxWKnnr: key sort work files for multiple key sorts running in parallel
  xxx is a number between 1 and the value that you specified for the MAXKSORT installation option or the OPTIONS MAXKSORT parameter. nn is a number between 1 and the value that you specified for SORTNUM installation option or the OPTIONS SORTNUM parameter.
- SYSOUunnr: sort message data sets
  unnn is a number between 1 and the value that you specified for the MAXKSORT installation option or the OPTIONS MAXKSORT parameter.
When you use dynamic allocation for these files, NGT Recover determines the optimal number of files to use.

For more information about MAXKSORT, see “MAXKSORT integer” on page 134, “MAXKSORT” on page 587, “Setting the MAXKSORT option” on page 479, and “Overview of the UNLOADKEYS and BUILDINDEX strategy” on page 463.

Sort work data sets for keys for non-parallel sorts

NGT Recover uses the following sort work data sets for non-parallel sorting of the index keys (when MAXKSORT=1):

- S001WK01 through S001WKnn for key sorts
- SYSOU001 for the sort message data set

For more information, see “Defining sort work data sets” on page 329.

Work data sets needed for extracted keys (WORKDDN)

If you code NOWORKDDN, no work data set is used for extracted keys.

If you code WORKDDN, you need to include a DD statement with the same name to specify the work data set. The default ddname is SYSUT1.

The work data set can be shared with other REBUILD INDEX command statements for other spaces. You must use the same work data set for all of the indexes on a single table space or code NOWORKDDN for all of them.

If you code neither WORKDDN nor NOWORKDDN, the use of a key work data set depends on whether you have included a SYSUT1 DD statement in the JCL. If you have included a SYSUT1 DD, it will specify the work data set. If you have not included a SYSUT1 DD, no work data set is used.

Note

WORKDDN is valid only if MAXKSORT=1. If MAXKSORT is a value greater than 1, processing occurs as if NOWORKDDN is coded.

DD statements for RECOVER UNLOADKEYS jobs

You must include DD statements in the JCL when you specify a RECOVER UNLOADKEYS command statement.
These DD statements are included in addition to those common to all NGT Recover executions (see “NGT Recover data sets and NGT Recover DD statements” on page 319). Use DD statements to specify the following data sets:

- Work data sets for extracted keys (SKEYDDN)
- (optional) Sort work data sets for keys

For examples of DD statements used in a combined RECOVER UNLOADKEYS and REBUILD INDEX job, see “Example 6: Extracting nonpartitioned index keys” on page 402.

**Work data set needed for extracted keys (SKEYDDN)**

This data set holds keys for subsequent RECOVER BUILDINDEX jobs. A sort for this file is done by RECOVER UNLOADKEYS. For the sorted extracted keys, one output data set exists per table space, regardless of how many partitions (or keys) are unloaded in one job.

The ddname is specified by the SKEYDDN option of a RECOVER UNLOADKEYS command. The default is SKEY.

When multiple RECOVER UNLOADKEYS commands are specified for a table space, the same SKEYDDN must be specified in all command statements.

The work data set for a specified table space cannot be shared with other RECOVER UNLOADKEYS commands for other table spaces or other REBUILD INDEX commands for other table spaces.

When you specify REBUILD INDEX and a RECOVER UNLOADKEYS for a table space, NOWORKDDN, which is the default value on the REBUILD INDEX command, sends keys for the partitioned index directly into the sort, bypassing the file held for the subsequent RECOVER BUILDINDEX job.

**Sort work data sets for keys**

These data sets are used for sorting the index keys. Sort work data sets are generally needed if the REBUILD INDEX command is included for the partitioned index. For more information, see “Defining sort work data sets” on page 329.

**DD statements for RECOVER BUILDINDEX jobs**

In addition to those common to all NGT Recover executions, DD statements must be included in the JCL when you specify a RECOVER BUILDINDEX command statement. Use DD statements for work data sets for extracted keys (SKEYDDN).

For examples of the DD statements used in a RECOVER BUILDINDEX job, see “Example 7: Building a nonpartitioned index” on page 402. For DD statements
common to all NGT Recover executions, see “DD statements common to all NGT Recover executions” on page 319.

**Data sets containing extracted keys (SKEYDDN)**

NGT Recover expects at least one DD statement that refers to a data set used to collect the sorted keys for nonpartitioned indexes in a prior RECOVER UNLOADKEYS run. If multiple files of sorted keys exist (because the RECOVER UNLOADKEYS command was run in multiple executions to get all of the partitions unloaded and sorted), each file needs a DD statement.

The ddname prefix is specified by the SKEYDDN option of a RECOVER BUILDINDEX command. The default prefix is SKEY. Any ddname within the JCL that begins with the specified prefix is recognized as an unload work data set.

**Defining sort work data sets**

You can define index sort work data sets in the following ways:

- Let the sort routine make the allocations dynamically by using installation defaults.
- Specify SORTDEVT and SORTNUM and let the sort routine dynamically make the allocations.
- Specify only SORTDEVT, and let BMCSORT determine the number of data sets.
- Code the DD statements in the JCL for the ddnames as follows:
  - L001WK01 through L001WKnn for log sorts (non-parallel)
  - LxxxWKnn for log sorts
    - **xxx** is the number of the log sort and is a number between 1 and the value that is specified for MAXLSORT. **nn** is the number of the work data set.
  - LOGOUunn sort message files
    - **unn** is the number of the log sort and is a number between 1 and the value that is specified for MAXLSORT.
  - S001WK01 through S001WKnn for key sorts (non-parallel)
  - SxxxWK01 through SxxxWKnn
    - **xxx** is a number between 1 and the value of the MAXKSORT option and **nn** is a number between 1 and the value specified for SORTNUM installation option or
the OPTIONS SORTNUM parameter, for key sort work files for multiple index key sorts running in parallel in subtasks.

Determining sort work space requirements

When a recovery involves sorting index keys or sorting the log, you should consider the following requirements:

- Virtual memory requirements, which are controlled by the SMCORE installation option (see “SMCORE=(0K,0K)” on page 591)

- Fixed disk storage (DASD) requirements, which are controlled by the following settings:
  - BMCSORT options settings (For more information, see the BMCSORT Reference Manual)
  - Parameter settings in the NGT Recover options module (AFR$OPTS), which are described in “NGT Recover installation options” on page 573

- Number of parallel index key sorts, which is controlled by the MAXKSORT option (“MAXKSORT integer” on page 134 and “MAXKSORT” on page 587) and the KSORTSHARE option (“MAXKSORT integer” on page 134 and “KSORTSHARE=YES” on page 586)

- Number of log sorts that can run concurrently

These requirements have an affect on sort performance. For more information, see “Managing sort performance” on page 474 and “Overview of the UNLOADKEYS and BUILDINDEX strategy” on page 463.

Index sort work space requirements

If you need to know the amount of sort work space that will be required for rebuilding indexes, you can request that NGT Recover provide this information. You do this by coding at least one sort work DD statement in your JCL.

For example, you could code the following DD statement:

```
S001WK01 DD DUMMY
```

You could then run NGT Recover with OPTIONS ANALYZE ONLY to receive a list of sort work data sets that will be used in this job along with the estimated space
required for each sort work data set. This output is in the AFRSTMT message data set. An example of the resulting output follows:

```
BMC96308I ESTIMATED NUMBER OF TRACKS FOR ALL INDEX KEY SORT WORK DATA SETS
FOLLOWS:
BMC96309I  S001WK**:       5627  (TRACKS)
BMC96309I  S002WK**:       5627  (TRACKS)
BMC96309I  S003WK**:       9184  (TRACKS)
BMC96309I  S004WK**:       9184  (TRACKS)
BMC96309I  S005WK**:       9184  (TRACKS)
BMC96309I  S006WK**:       5627  (TRACKS)
```

You can then code DD statements for one or more of the sort work data sets with the amount of disk space given for each data set in the AFRSTMT output. If you do not code DD statements for some of the sort work data sets, BMCSORT dynamically allocates them.

**Note**

NGT Recover provides an approximation only for the amount of sort work spaces required even if statistics or NUMRECs are accurate. Real space requirements could vary depending on amount of memory available at run time, number and size of sorts running in parallel, the methods of sorts chosen by BMCSORT, and other variables.

**WARNING**

You should use the information generated by the technique described above with caution. If there were significant changes in recoverable objects, used options, environment, and other variables between the ANALYZE ONLY run and real recovery run, NGT Recover could change the recovery plan.

If you do not code any sort work DDs in your JCL (you allow BMCSORT to dynamically allocate all sort work data sets), NGT Recover dynamically selects work files for each sort based on availability, and which set of sort work files are assigned to a particular index is not predictable. In this case, the information in AFRSTMT, as shown in the example output above, is only useful in determining the total amount of required sort work space.

If you code at least one sort work DD in your JCL, NGT Recover statically assigns sort work data sets to indexes during the ANALYZE phase. In this case, you can reliably determine the space requirements for each set of sort work data sets from AFRSTMT, as shown in the example output above.

The estimates that NGT Recover provides are based entirely on the estimated number of keys for each index. NGT Recover obtains this information from the NUMREC parameter if you specify it on the REBUILD statement. Otherwise, NGT Recover uses the following values to determine the estimates:

- For DB2 Version 10 and later with the UT SORT DATA SET ALLOCATION field (DSN6SPRM UTSORTAL subsystem parameter) set to NO (the default value),
NGT Recover uses partition or table cardinality if a RUNSTATS operation has been performed for the table space.

- For DB2 Version 10 and later with UTSORTAL set to YES, NGT Recover uses real-time statistics by using the TOTALROWS value from table SYSIBM.SYSTABLESPACESTATS for single-table table spaces.

**Note**

If UTSORTAL is set to YES and you perform a migration, TOTALROWS may not be accurate. If it is not accurate after the migration, you must reset the value to NULL. However, TOTALROWS is generally accurate and you may not need to reset its value in the following situations:
- If you migrate data daily and the size of the space does not change much
- If you include the real-time statistics tables in your migration

If none of these sources is available, NGT Recover attempts to calculate the number of keys by using the size of the table space data sets. This calculation can be very inaccurate if the table space contains multiple tables or the table contain VARCHAR columns. To get accurate sort work size estimates, you should

- Ensure that RUNSTATS is run frequently
- Use real-time statistics (by setting UTSORTAL to YES)
- Provide an accurate estimate on the NUMREC parameter

The amount of parallelism that NGT Recover can achieve can be affected by coding sort work DDs in the JCL. When you code one or more sort work DDs, NGT Recover statically assigns sorts to indexes, which prevents any overlap of index rebuilds with table space recoveries if keys are being extracted.

For example, assume that you want to recover table space A and table space B and rebuild all indexes for the two table spaces (and MAXLSORT=1). If table space A needs 8 sorts, table space B needs 12 sorts, and MAXKSORT is set to 20, NGT Recover can overlap the building of indexes for table space A with the recovery of table space B. Table space A will use sort work data sets S001WK** through S008WK** and table space B will use sort work data sets S009WK** through S020WK**.

Using this same example, if you code any sort work DDs in your JCL, NGT Recover will use S001WK** through S008WK** for table space A and S001WK** through S012WK** for table space B. When the recovery for table space B begins, the recovery will have to wait for the index builds for table space A to complete because sort work data sets S001WK** through S008WK** are not available.
**Note**

BMC recommends that you allow BMCSORT to dynamically allocate all sort work data sets. You should code sort work DD statements in your JCL only if you have a special situation where dynamic allocation will not work.

---

**Log sort space requirements**

If a significant number of log records exist, the sort process can require additional system resources.

NGT Recover invokes BMCSORT. The amount and type of resources that are required for the sort are dependent on the number of log records to be sorted. The total amount of sort work space that is needed to sort the log optimally is roughly equal to the sum of the lengths of all log records involved in the recovery times two. You can specify L001WKnn DD statements in the JCL for log sort work data sets or use the SORTDEVT and SORTNUM options for NGT Recover.

You can use the MAXLSORT option ("MAXLSORT integer" on page 136 and "MAXLSORT=0" on page 588) to specify that you want to run multiple log sorts.

You can specify LxxxWKnn DD statements in the JCL for log sort work data sets or use the SORTDEVT and SORTNUM options for NGT Recover.

The LOGSCAN command can help you size the log sort work data sets by estimating the number of bytes of log data to be sorted. This command also provides NUMREC and AVGRECSZ estimates that can be used to improve log sort performance.

**Note**

The sorting requirement at recovery time can be reduced significantly by running R+/CHANGE ACCUM in advance. For more information, see the R+/CHANGE ACCUM for DB2 User Guide.

---

**Determining unloaded keys file space requirements**

Use the following information to help estimate the size to allocate for unloaded keys files. Your RUNSTATS must be current, or you must use your own row estimate.

Multiply the number of rows in the table space by the record length to get the bytes needed by NGT Recover, as follows:

\[
\text{Size} = NR \times RL
\]
In the equation, the variables are defined as follows:

- **Size** is bytes in unloaded keys file.
- **NR** is the number of rows given by the value of CARD from the SYSIBM.SYSTABLEPART table.
- **RL** is the record length displayed in message BMC40806 LENGTH OF RECORDS=length. This message prints in the NGT Recover plan when SKEYDDN is specified and can be obtained from an ANALYZE ONLY job. This length is the maximum key size involved plus six or seven bytes. (For more details, see “Index sort work space requirements” on page 330.)

For unloaded keys files, all partitions in a run are added together.

### Specifying NGT Recover syntax

Place recovery commands in the SYSIN data set.

Use “NGT Recover syntax” on page 83 and “Examples of NGT Recover jobs” on page 367 as guides to the syntax that is required for completion of your job setup.

### Running NGT Recover jobs

Running the NGT Recover utility may include the following tasks:

- Starting NGT Recover as a batch job
- Restarting NGT Recover from the last phase or sync point
- Displaying the status of an NGT Recover job
- Terminating a NGT Recover job

To run an NGT Recover job, you must have the proper authorizations. For more information, see “Authorization” on page 68.

If an NGT Recover job abnormally terminates, you must clean up the BMCUTIL and BMCSYNC tables as described in “Cleaning up the BMCUTIL and BMCSYNC tables” on page 340. A RECOVER UNLOADKEYS job also requires some cleanup as discussed in “Cleaning up after a RECOVER UNLOADKEYS job” on page 341.
Starting an NGT Recover job

You normally invoke NGT Recover as a batch job by specifying execution of the module AFRMAIN on the EXEC statement of your JCL, with required utility parameters.

Some DD statements may also be necessary. For more information, see “NGT Recover data sets and NGT Recover DD statements” on page 319.

If you are not using SIMULATE YES option or the NGT Recover redirection option, INDEP OUTSPACE, the table spaces or indexes that are specified in your NGT Recover commands must have an acceptable status before the job will run. For information about unacceptable table and index space status, see “Initial table space and index space status” on page 71.

Restarting an NGT Recover job

You can restart an NGT Recover job that fails to complete successfully. If checkpoints were taken in the original run, processing phases will not be executed again during the restart.

The value that you specify for the restart parameter ("Restart parameter (restartParm)" on page 314) controls the restarting of the utility. You can restart the utility from the last recorded checkpoint by specifying NEW/RESTART or RESTART for the restart parameter value.

If a table space, index, or index space was marked unrecoverable because of unavailable recovery resources, NGT Recover does not attempt to recover that object in a restarted job. If the recovery resources have been made available since the original job, the objects should be recovered by using the NEW restart parameter.

---

**Note**

When you perform a RESTART, NGT Recover does not attempt to recover unavailable resources, but NGT Recover does issue messages in the planning phase indicating that the resources are not there and gives the appropriate return code.

---

The following table shows the kinds of restart that are possible for each phase of NGT Recover.

**Table 11: Restarts allowed for each phase of NGT Recover**

<table>
<thead>
<tr>
<th>Last incomplete phase</th>
<th>Phase restart allowed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILINIT/ANALYZE</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Chapter 4 Building and running NGT Recover jobs 335
<table>
<thead>
<tr>
<th>Last incomplete phase</th>
<th>Phase restart allowed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG INPUT</td>
<td>Yes</td>
</tr>
<tr>
<td>SNAP</td>
<td>Yes</td>
</tr>
<tr>
<td>MERGE&lt;sup&gt;a&lt;/sup&gt;</td>
<td>No</td>
</tr>
<tr>
<td>RESTORE</td>
<td>Yes</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>Yes</td>
</tr>
<tr>
<td>BUILD&lt;sup&gt;b&lt;/sup&gt;</td>
<td>No</td>
</tr>
<tr>
<td>UTILTERM</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<sup>a</sup> If you specify restart for a job that failed in a MERGE phase, restart may occur at the beginning of the LOG INPUT phase. However, recoverable objects that were successfully merged will not be remerged during the restart job. True phase restart at such a MERGE phase is possible only for a merge with no log records to be sorted.

<sup>b</sup> A phase restart from the BUILD phase is possible only when you specify WORKDDN with REBUILD INDEX. If you specify NOWORKDDN, the restarted run performs an UNLOAD phase, and then a BUILD phase.

To restart a failed job successfully, you must use the same utility ID as in the original run. Also, you must ensure that the commands in the SYSIN data set are compatible with those in the original job. For example, you can remove objects (table spaces and indexes) from the job, but you cannot add objects, nor can you change critical attributes of the job or its objects. The values for ANALYZE must remain the same. None of the index commands can be changed to another type. For example, REBUILD INDEX cannot be changed to RECOVER UNLOADKEYS. Ensure that none of the objects involved in the recovery have been altered between the time the original job terminated and the time of the restart.

The following must apply in the original and restarted jobs:

- For each RECOVER TABLESPACE, RECOVER INDEXSPACE, and RECOVER INDEX command in the job, LOGAPPLY ONLY, LOGONLY, FROMRBA/FROMLOGPOINT, TORBA/TOLOGPOINT, TOCOPY, and INDEP OUTSPACE (and the associated data sets) must remain the same.

- For each REBUILD INDEX command, the values of WORKDDN (or NOWORKDDN) and INDEP OUTSPACE (and the associated data sets) must remain the same.

- For RECOVER UNLOADKEYS commands, the values of PART and SKEYDDN must remain the same.

- For the RECOVER BUILDINDEX commands, the value of SKEYDDN must remain the same.
To ensure restartability during the BUILD phase for rebuilding an index, specify DISP=(MOD,CATLG,CATLG) in the DD statement for each key work data set. If you must restart the run, the key work data set will be referred to as DISP=OLD by the program. This DISP value results in correct handling during restart. A subsequent step can delete the work data sets conditionally.

**Note**
You cannot restart an NGT Recover job that specifies the ANALYZE ONLY option or the SIMULATE YES option. You must start a new job.

### Restart considerations when creating output copies

Whether you restart your NGT Recover jobs from the beginning or restart from the point of failure plays a role in how your restart jobs should be set up.

#### Restart from the beginning

If you always plan to restart failed NGT Recover jobs from the beginning, use the NEW restart parameter in the original job. If you allocate the output copy data sets in the JCL, you must also do the following things:

- Use DISP=(NEW, CATLG, CATLG) or DISP=(NEW, KEEP, KEEP)
- Use unique data set names for each execution
  - GDGs are helpful for accomplishing this task.
- Code BLKSIZE=0 for disk data sets so that unopened data sets can be migrated successfully, if necessary

This method leaves empty, unused data sets if disk copies are made for any object not recovered by the failing run.

The advantage of this method is that only the data set names must be modified for the restart. If GDGs are used, however, no modification of the JCL is necessary. Do not use this method under a restart package that modifies the data set names or dispositions on restart.

#### Restart from the point of failure

If you plan always to restart failed NGT Recover jobs from the point of failure, use the NEW/RESTART restart parameter in the original job. If you allocate copy data sets in the JCL, you must also

- Use DISP=(MOD, CATLG, CATLG) or DISP=(NEW, KEEP, KEEP)
- Perform the following additional steps at restart:
— For stacked tape copies that use GDGs, modify the data set names to indicate the generation relative to the restart. Modify the (+ n) value to (+ n- m), where m is the relative generation number for the last cataloged generation in the original execution.

— For cataloged stacked tape copies, remove VOL=REF= from the copy data set DD statements for the copy creation that failed. This removal tells the system to use the catalog for volume information.

— Failure to remove VOL=REF= causes the restarted data set to get a "not cataloged" message and causes a multiple volume data set to be on a different set of volumes than the original, cataloged data set. If the restarted copy data sets expand to more volumes than were cataloged at the time of the original execution, any attempt to stack further data sets by using VOL=REF= results in another abend because the reference uses the catalog information from the beginning of the job step. The system will catalog the expanded data sets again at the end of the job step. However, submitting the job a third time should result in the utility executing with the volumes resolved correctly.

— For uncataloged stacked tape copies, you must include the VOL=SER information of completed copies in the DD statements before restarting. You must also change the NEW disposition to OLD.

This method allows processing to continue with the failed command, minimizing unnecessary processing. This method usually requires manual intervention during restart to modify the JCL.

**Note**

The use of the output copy dynamic allocation feature eliminates the need to modify the JCL during restart. (For more information, see “Allocating output image copy data sets dynamically” on page 491.)

### Restart considerations when using TRANSFORM

If the job fails performing an index transformation, the restart will reprocess the entire table space along with the index, even if the table space transformation completed successfully.

If the target table space is a partitioned-by-range (PBR) space and the job fails while processing a partition, the restart will not reprocess the partitions that have completed successfully unless the indexes are also being transformed. In this case, the restart will reprocess all the table space partitions. If the target table space is a partition-by-growth (PBG) space, the restart will reprocess the entire table space.
NGT Recover installation options that affect restart—CHECKPT and CHECKINT

Two installation options that affect the restart processing for NGT Recover jobs are CHECKPT and CHECKINT. These options are provided to minimize the time that is lost redoing work when a job fails and must be restarted.

The CHECKPT and CHECKINT installation options are described in “NGT Recover installation options” on page 573.

You can specify the CHECKPT installation option to take the following checkpoints:

- Take no checkpoints
- Take checkpoints only at the end of each phase

You can override the CHECKPT installation option at run time by specifying the NGT Recover checkpoint override parameter that is described on “Checkpoint override parameter (checkPt)” on page 316.

Note

If no checkpoints are requested in the installation option, NGT Recover will not take checkpoints, except those necessary to synchronize NGT Recover execution with the execution of other BMC utilities, as well as MERGE checkpoints that are necessary to guarantee the integrity of output copy, point-in-time recovery, or index rebuild registrations.

When you request checkpoints, you can prevent unnecessary checkpoints by specifying a value for the checkpoint interval installation option, CHECKINT, which sets the number of minutes elapsed between checkpoints. You can override the CHECKINT installation option at run time by specifying CHECKINT on the OPTIONS command, as described in “CHECKINT (integer)” on page 112.

To determine values for either or both of these options, you must balance the cost of taking checkpoints against the time that is lost redoing work when an NGT Recover execution must be restarted.

Restart considerations with INDEXLOG AUTO

When NGT Recover converts a RECOVER INDEX request to a REBUILD INDEX request based on the specification of INDEXLOG AUTO, NGT Recover updates the BMCSYNC table for the index to indicate that it is now a REBUILD INDEX. During restart, NGT Recover uses the BMCSYNC table update to discover that an index recovery was converted.
If a RECOVER INDEX request that uses an image copy fails during execution, the restart logic causes the index to become a candidate for conversion because the original image copy is invalid.

If a RECOVER INDEX DSNUM ALL request for a nonpartitioned index is converted to multiple DSNUM n requests, and if one of the resulting MERGE phases later fails, restart does not allow the request to be converted to a REBUILD INDEX request. If a multi-data-set index is recovered with a DSNUM 0 image copy, the RECOVER INDEX request can convert to a REBUILD INDEX request.

**Displaying the status of an NGT Recover job**

You can determine the status of NGT Recover jobs in progress or awaiting restart by issuing an SQL SELECT statement on the BMCUTIL table.

For a sample statement, see “To display BMC utility status” on page 609.

**Terminating an NGT Recover job**

You can terminate an active recovery job by deleting the corresponding row from the BMCUTIL table.

For a description of the BMCUTIL table, see “BMCUTIL table” on page 623.

If the job is in progress, it terminates at the next checkpoint with a return code of 8.

If you require immediate termination, you must cancel the job with the operating system CANCEL command.

If the NGT Recover utility is terminated during any phase except UTILINIT, ANALYZE, or UTILTERM, the table spaces or indexes being recovered may be unusable. When NGT Recover terminates abnormally, spaces are left in STOP (stopped) status. However, spaces that are not recovered may be left in RECP (recovery pending) status, RBDP (rebuild pending) status, PSRCP (page set recovery pending) status, or PSRBD (page set rebuild) status.

**Cleaning up the BMCUTIL and BMCSYNC tables**

You can clean up after an NGT Recover run terminates abnormally by executing NGT Recover with the same utility ID and specifying TERM as the value of the restart parameter.
This cleanup deletes all rows for the utility ID in the BMCUTIL and BMCSYNC tables.

Cleaning up after a RECOVER UNLOADKEYS job

Successful completion of a RECOVER UNLOADKEYS job leaves rows in the BMCSYNC table with blank utility IDs for the table space partitions and the indexes for which keys are being unloaded.

If you decide not to run RECOVER BUILDINDEX after a RECOVER UNLOADKEYS, you can remove these rows with the SQL DELETE statement in “Cleaning up RECOVER UNLOADKEYS entries” on page 619.

Using MSGLEVEL to control NGT Recover output

You can control the amount of output that NGT Recover produces based on the value that you specify for the MSGLEVEL utility parameter.

Valid values of MSGLEVEL are 0, 1, and 2, with a default value of 1 (see “Message level parameter (msgLevel)” on page 316). The example in the following figure was run by using each value of MSGLEVEL. Different output files are received, and different information is in the output files, for each run. The files that are produced are based on the different values of MSGLEVEL as summarized in Table 9 on page 320.

Figure 55: NGT Recover JCL for recovery of table space and index

//AFREXMLx JOB (PAFR),'EXAMPLE MSGVLx',
//CLASS=Q,NOTIFY=&SYSUID,
//MSGCLASS=X
/*/ 
** THIS JOB RECOVERS A TABLE SPACE AND REBUILDS THE INDEX TO CURRENT.
*/ 
** EXAMPLE OF MSGLEVEL(x) OUTPUT.
/*/ 
** 
RECOVER EXEC PGM=AFRMAIN,REGION=0M, 
PARM='DGE,AFRMSGLx,NEW/RESTART,MSGLEVEL(x)' <== x set to 0, 1, or 2 
STEPLIB DD DISP=SHR,DSN=product.libraries 
   DD DISP=SHR,DSN=DB2.DSNEXIT 
   DD DISP=SHR,DSN=DB2.DSNLOAD 
SYSIN DD *
OPTION EARLYRECALL INDEXLOG NO 
   RECOVER TABLESPACE BMCDBSMP.BMCTS014 
   REBUILD INDEX (RMD.BMCIX014) 
*/ 
//
Sample output with MSGLEVEL(0)

When you specify MSGLEVEL(0), NGT Recover produces the following output files:

- AFRSUMRY (Figure 56 on page 342)
- AFRSTMT (Figure 57 on page 342)
- AFRPR001 (Figure 58 on page 345)

Figure 56: AFRSUMRY with MSGLEVEL(0)

<table>
<thead>
<tr>
<th>BMC40018I</th>
<th>NGT RECOVER FOR DB2 ZOS V12.1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40473I</td>
<td>COPYRIGHT BMC SOFTWARE INC. 1991-2015</td>
</tr>
<tr>
<td>BMC96173I</td>
<td>RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001</td>
</tr>
<tr>
<td>BMC40474I</td>
<td>CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO <a href="mailto:SUPPORT@BMC.COM">SUPPORT@BMC.COM</a></td>
</tr>
</tbody>
</table>

BMC40001I UTILITY EXECUTION STARTING 08/28/2014 10:45:51

BMC40018I
BMC40876I MAINT: NO RECOVER PLUS PTFS APPLIED

BMC96010I SOLUTION COMMON CODE V11.2.01
BMC40876I MAINT: NO SOLUTION COMMON CODE PTFS APPLIED

BMC96010I BMCSORT ENGINE V2.3.01
BMC40876I MAINT: BPJ0195 BPJ0198 BPJ0209 BPJ0250 BPJ0263 BPJ0269 BPJ0279
BMC40876I MAINT: BPJ0288 BPJ0308 BPJ0343 BPJ0357 BPJ0361 BPJ0374 BPJ0390
BMC40876I MAINT: BPJ0404 BPJ0473 BPJ0503 BPJ0521

BMC40002I UTILITY ID = EXAMPLE11, DB2 SUBSYSTEM ID = DEFQ.
BMC40094I ATTEMPTING TO CONNECT TO DB2 SUBSYSTEM DEFQ, USING PLAN AFRB111T
BMC40024I SUCCESSFUL CONNECT TO DEFQ (RELEASE 1010 OF DB2) USING PLAN AFRB111T

BMC40475I INPUT STATEMENTS PRINTED IN AFRSTMT

BMC40336I ANALYZE FINISHED.
BMC40287I ELAPSED TIME = 00:00:03, UTILITY ELAPSED TIME = 00:00:04
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I PLANNING FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:05
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I PLAN EXECUTION FINISHED.
BMC40287I ELAPSED TIME = 00:00:01, UTILITY ELAPSED TIME = 00:00:07
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40476I EXECUTION SUMMARY
BMC40477I TABLE SPACES / PHASES (PARTITIONS) RECOVERED : 1 (0)
BMC40479I LOG RECORDS APPLIED : 24995
BMC40480I INDEXES REBUILT : 1

BMC40005I UTILITY EXECUTION COMPLETE 08/28/2014 10:45:58, RETURN CODE = 0

Figure 57: AFRSTMT with MSGLEVEL(0)

<table>
<thead>
<tr>
<th>BMC40018I</th>
<th>NGT RECOVER FOR DB2 ZOS V12.1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40473I</td>
<td>COPYRIGHT BMC SOFTWARE INC. 1991-2015</td>
</tr>
<tr>
<td>BMC96173I</td>
<td>RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001</td>
</tr>
<tr>
<td>BMC40474I</td>
<td>CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO <a href="mailto:SUPPORT@BMC.COM">SUPPORT@BMC.COM</a></td>
</tr>
</tbody>
</table>

BMC40931I PARM LIST: DB2 SSID = DEFQ
Using MSGLEVEL to control NGT Recover output

Chapter 4  Building and running NGT Recover jobs    343
BMC409671 TAPE DEVICE04 = ATLVT5
BMC409671 TAPE DEVICE05 = ATL3592
BMC409671 TAPE DEVICE06 = CART
BMC409671 TAPE DEVICE07 = CARTVTS
BMC409671 TAPE DEVICE08 = CRT8NS
BMC409671 TAPE DEVICE09 = CRT9NS
BMC409671 TAPE DEVICE10 = DANBARR
BMC409671 TAPE DEVICE11 = EXT3592
BMC409671 TAPE DEVICE12 = IBMATL
BMC409671 TAPE DEVICE13 = IBMVTS
BMC409671 TAPE DEVICE14 = PHXATL
BMC409671 TAPE DEVICE15 = SA3590
BMC409671 TAPE DEVICE16 = TAPEABL
BMC409671 TAPE DEVICE17 = 3590B
BMC409671 TAPE DEVICE18 = 3590E
BMC409671 TAPE DEVICE19 = 3480
BMC409671 TAPE DEVICE20 = 3400-9
BMC409671 TAPE DEVICE21 = SYS3480R
BMC409671 TAPE DEVICE22 = SYS348XR

BMC40018I NGT RECOVER FOR DB2 ZOS V12.1.00
BMC40876I MAINT:  NO RECOVER PLUS PTFS APPLIED

BMC40068I THE KEYS FOR INDEX RMD.BMCIX014 WILL GO DIRECTLY TO THE SORT BECAUSE THERE IS NO SYSUT1 DD STATEMENT IN THE JCL

BMC40335I UTILID EXAMPLE11 WAS NOT FOUND IN THE BMCUTIL TABLE - UTILITY WILL RUN AS IF 'NEW' WERE SPECIFIED

BMC40960I INDEXLOG STRATEGY DISABLED

BMC40936I USING STANDARD DB2 SECURITY

BMC40335I SYNCHRONIZATION STARTING 08/28/2014 10:45:53
BMC40336I SYNCHRONIZATION FINISHED.

BMC96306I MAXLSORT REDUCED FROM 50 TO 17 DUE TO MEMORY CONSTRAINTS
Sample output with MSGLEVEL(1)

When you specify MSGLEVEL(1), NGT Recover produces the following output files:

- AFRSUMRY, which is identical to the output of MSGLEVEL(0) (Figure 56 on page 342)
- AFRSTMT, which is similar to the output with MSGLEVEL(0) (Figure 57 on page 342)
- AFRPRINT (Figure 59 on page 346)
- AFRPR001 (Figure 60 on page 346)
- AFROSUM (Figure 61 on page 348)

Figure 59: AFRPRINT with MSGLEVEL(1)

```
BMC40018I RECOVER PLUS FOR DB2 V11.1.00   (RECOVER AFRPRINT)
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM
BMC96401I LOG INPUT PHASE MESSAGES ARE IN FILE AFRPR001
BMC96400I MERGE PHASE MESSAGES FOR THE FOLLOWING OBJECT ARE IN FILE AFRPR001
BMC40301I TABLESPACE = BMCDBSM.BMCTS014
BMC40303I INDEX = RMD.BMCIX014
```

Figure 60: AFRPR001 with MSGLEVEL(1)

```
BMC40018I RECOVER PLUS FOR DB2 V11.1.00   (RECOVER AFRPRINT)
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM
BMC40016I LOG INPUT PHASE STARTING 08/28/2012 10:46:46
BMC40954I AVAILABLE MEMORY BELOW THE LINE = 10044K, TOTAL AVAILABLE MEMORY (AT LEAST) = 1370496K
BMC40441I BMCSORT PRODUCT LOCATED AND WILL BE USED
BMC40020I SYSEVENT STGTEST STORAGE AVAILABLE = 5606844
BMC40020I SORT FILE SIZE = 2448384, SORT MEMORY = 242407
BMC40725I PROCESSING LOG RANGE:
BMC40302I DSN = DEFQCAT.LOGCOPY1.DS11.DATA
BMC40193I 3 LOG DATASET BUFFERS OF 737280 BYTES EACH WERE ALLOCATED
BMC40302I DSN = DEFQCAT.LOGCOPY1.DS11.DATA
BMC40117I **********     PERFORMANCE STATISTICS FROM LOG READ STEP     **********
BMC40120I 1 FILES READ     0 I/O WAITS        I/O WAIT TIME = 00:00:00
BMC40124I 3 SORT WAITS      SORT WAIT TIME = 00:00:00
BMC40172I 24995 LOG RECORDS AND 1759648 BYTES OF LOG DATA PROCESSED
BMC40302I DSN = DEFQCAT.LOGCOPY1.DS11.DATA
BMC40173I TIME SPENT IN DATASET ALLOCATION = 00:00:00
BMC40174I 0 I/O WAITS     I/O WAIT TIME = 00:00:00
BMC40172I 24995 LOG RECORDS AND 1759648 BYTES OF LOG DATA PROCESSED
BMC40012I LOG INPUT PHASE COMPLETE. ELAPSED TIME = 00:00:00, TIME SINCE UTILITY START = 00:00:02
BMC40868I LOG INPUT PHASE COMPLETE. ACCUMULATED TCB TIME = 00:00:00
```
Chapter 4 Building and running NGT Recover jobs 347
Using MSGLEVEL to control NGT Recover output

Figure 61: AFROSUM with MSGLEVEL(1)
Using MSGLEVEL to control NGT Recover output

---

**Sample output with MSGLEVEL(2)**

When you specify MSGLEVEL(2), NGT Recover produces the following output files:

- **AFRSUMRY**, which is identical to the output of MSGLEVEL(0) (Figure 56 on page 342)
- **AFRSTMT** (Figure 62 on page 349)
- **AFRPRINT** and **AFRPR001** (Figure 63 on page 353 and Figure 64 on page 353)
- **AFROSUM**, which is identical to the output of MSGLEVEL(1) (Figure 61 on page 348)
- **AFRPLAN** (Figure 65 on page 356)

*Figure 62: AFRSTMT with MSGLEVEL(2)*

---
Using MSGLEVEL to control NGT Recover output
Using MSGLEVEL to control NGT Recover output

Chapter 4 Building and running NGT Recover jobs

BMC40967I TAPE DEVICE16 = TAPEABL
BMC40967I TAPE DEVICE17 = 3590B
BMC40967I TAPE DEVICE18 = 3590E
BMC40967I TAPE DEVICE19 = 3480
BMC40967I TAPE DEVICE20 = 3400-9
BMC40967I TAPE DEVICE21 = SYS3480R
BMC40967I TAPE DEVICE22 = SYS348XR

BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40876I MAINT: NO RECOVER PLUS PTFS APPLIED

BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40876I MAINT: NO RECOVER PLUS PTFS APPLIED

BMC96010I SOLUTION COMMON CODE V11.2.01
BMC40876I MAINT: NO SOLUTION COMMON CODE PTFS APPLIED

BMC96010I BMCSORT ENGINE V2.3.01
BMC40876I MAINT: BPJ0195 BPJ0198 BPJ0209 BPJ0250 BPJ0263 BPJ0269 BPJ0279
BMC40876I MAINT: BPJ0288 BPJ0308 BPJ0343 BPJ0357 BPJ0361 BPJ0374 BPJ0390
BMC40876I MAINT: BPJ0404 BPJ0473 BPJ0503 BPJ0521

BMC40335I PACKAGE CHECKING AND BINDING STARTING 08/28/2014 10:47:23
BMC40336I PACKAGE CHECKING AND BINDING FINISHED.

BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40020I EBCDIC CCSID = 37
BMC40037I UTILID EXAMPML2 WAS NOT FOUND IN THE BMCUTIL TABLE - UTILITY WILL RUN AS IF 'NEW' WERE SPECIFIED

BMC40101I OPTION EARLYRECALL INDEXLOG NO
BMC40101I RECOVER TABLESPACE BMCDBSMP.BMCTS014
BMC40101I REBUILD INDEX (RMD.BMCIX014)
BMC40101I

BMC40960I INDEXLOG STRATEGY DISABLED
BMC40964I ZIP ENABLED (0) USING SUBSYSTEM ACP1
BMC40971I z/OS RELEASE = 1.13.00, PID = HBB77B0, SMS RELEASE = 1.13.0
BMC40157I AVAILABLE REGION BELOW 16M = 10056K, AVAILABLE REGION ABOVE 16M = 1362336K, NUMBER OF CPUS = 5
BMC40020I BYT:0, 0, 0; BLK:0, 0, 0; TOT:298932, 5608020, 5608020; iter: 1; RC:4

BMC40335I ANALYZE STARTING 08/28/2014 10:47:23
BMC40336I PROC RO STMT FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I PROC IX STMT FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSTB FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSIX FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSKY FINISHED.
Using MSGLEVEL to control NGT Recover output

BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  SCAN SYSFL FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC400681  THE KEYS FOR INDEX RMD.BMCIX014 WILL GO DIRECTLY TO THE SORT BECAUSE THERE IS NO SYSUT1 DD STATEMENT IN THE JCL
BMC403361  TS LOOKUP FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC406331  USING STANDARD DB2 SECURITY
BMC403361  DB LOOKUP FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  TB LOOKUP FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  IP LOOKUP FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  SG LOOKUP FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC407051  VOLUMES IN STOGROUP SYSDEFLT WILL BE USED IN THE FOLLOWING ORDER:
BMC407071   (DEVXXX)
BMC403361  SCAN SYSVOL FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  CHK DS FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  HDR PG RD FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  SYSCOPY LOOKUP FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403361  CATALOG LOOKUP FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC403351  SYNCHRONIZATION STARTING 08/28/2014 10:47:24
BMC400251  OUTPUT FROM COMMAND '-STO DB(BMCDBSMP) SPACE(BMCTS014) AT (COMMIT)' FOLLOWS:
BMC400201  DSN9022I  *DEFQ DSNTDDIS 'STOP DATABASE' NORMAL COMPLETION
BMC400251  OUTPUT FROM COMMAND '-STO DB(BMCDBSMP) SPACE(BMCTS014) AT (COMMIT)' FOLLOWS:
BMC400201  DSN9022I  *DEFQ DSNTDDIS 'STOP DATABASE' NORMAL COMPLETION
BMC403361  SYNCHRONIZATION FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00

BMC963061  MAXLSORT REDUCED FROM 50 TO 17 DUE TO MEMORY CONSTRAINTS
BMC403361  DDSCAN FINISHED.
BMC402871  ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC402881  CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00
LOG FILE RESOURCES:
DSN = DEFQCAT.LOGCOPY1.DS11.DATA
USING LOG DEFQCAT.LOGCOPY1.DS11.DATA FOR PROCESSING
FROMRBA = X'00000000005917EE13DD00' TORBA = X'00000000005917F068756'

LOG INPUT PHASE MESSAGES ARE IN FILE AFRPR001
Merging phase messages for the following object are in file AFRPR001
Tablespace = BMCDBSMP.BMCTS014

BUILD PHASE MESSAGES FOR THE FOLLOWING OBJECT ARE IN FILE AFRPR001
Index = RMD.BMCI014

LOG INPUT PHASE STARTING 08/28/2014 10:47:25

Figure 63: AFRPRINT with MSGLEVEL(2)

Figure 64: AFRPR001 with MSGLEVEL(2)
BMC40124I 3 SORT WAITS SORT WAIT TIME = 00:00:00
BMC40172I 24995 LOG RECORDS AND 1759648 BYTES OF LOG DATA PROCESSED
BMC40302I DSN = DEFQCAT.LOGCOPY1.DS11.DATA
BMC40173I TIME SPENT IN DATASET ALLOCATION = 00:00:00
BMC40174I 0 I/O WAITS I/O WAIT TIME = 00:00:00
BMC40172I 24995 LOG RECORDS AND 1759648 BYTES OF LOG DATA PROCESSED
BMC40012I LOG INPUT PHASE COMPLETE. ELAPSED TIME = 00:00:00, TIME SINCE UTILITY START = 00:00:02
BMC40016I MERGE PHASE STARTING 08/28/2014 10:47:25
BMC40301I TABLESPACE = BMCDBSMP.BMCTS014
BMC40484I MESSAGES FROM PHASE SUBTASK:
BMC40060I PENDING FLAG RECP SET FOR
BMC40345I SPACE = BMCDBSMP.BMCTS014
BMC40356I AN IDCAMS 'DELETE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET
DEFQCAT.DSNDBC.BMCDSPM.BMCTS014.I0001.A001
BMC40356I AN IDCAMS 'DEFINE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET
DEFQCAT.DSNDBC.BMCDSPM.BMCTS014.I0001.A001
BMC40350I DATA SET DEFQCAT.DSNDBD.BMCDSPM.BMCTS014.I0001.A001 WAS SUCCESSFULLY
ALLOCATED TO DDNAME SYS00007 ELAPSED = 00:00:00
BMC40485I END OF PHASE SUBTASK MESSAGES
BMC40793I EXTRACTING KEYS FOR:
BMC40303I INDEX = RMD.BMCIX014
BMC40350I DATA SET AFR.BMCDSPM.BMCTS014.IC.G0003V00 WAS SUCCESSFULLY ALLOCATED TO
DDNAME SYS00007 ELAPSED = 00:00:00
BMC40868I LOG INPUT PHASE COMPLETE. ACCUMULATED TCB TIME = 00:00:00
BMC40016I MERGE PHASE STARTING 08/28/2014 10:47:25
BMC40301I TABLESPACE = BMCDBSMP.BMCTS014
BMC40484I MESSAGES FROM PHASE SUBTASK:
BMC40793I EXTRACTING KEYS FOR:
BMC40303I INDEX = RMD.BMCIX014
BMC40350I DATA SET DEFQCAT.LOGCOPY1.DS11.DATA WAS SUCCESSFULLY ALLOCATED TO DDNAME
SYS00004 ELAPSED = 00:00:00
BMC40485I END OF PHASE SUBTASK MESSAGES
BMC40793I EXTRACTING KEYS FOR:
BMC40303I INDEX = RMD.BMCIX014
BMC40350I DATA SET AFR.BMCDSPM.BMCTS014.IC.G0003V00 WAS SUCCESSFULLY ALLOCATED TO
DDNAME SYS00007 ELAPSED = 00:00:00
BMC40793I EXTRACTING KEYS FOR:
BMC40303I INDEX = RMD.BMCIX014
BMC40350I DATA SET AFR.BMCDSPM.BMCTS014.IC.G0003V00 WAS SUCCESSFULLY ALLOCATED TO
DDNAME SYS00007 ELAPSED = 00:00:00
BMC40906I NUMBER OF PAGES EXAMINED = 4999, ROWS EXAMINED = 4999, KEYS EXTRACTED = 4999
BMC40907I INDIRECT REFERENCES: FORWARD = 0, BACKWARD = 0
BMC40902I TOTAL WAITS = 16, TOTAL WAIT TIME = 00:00:00
BMC40979I LOG INPUT WAITS = 1, WAIT TIME = 00:00:00
BMC40981I FULL IC AFR.BMCDSPM.BMCTS014.IC.G0003V00 READ WAITS = 1, WAIT TIME = 00:00:00
BMC40902I SPACE WRITE WAITS = 14, WAIT TIME = 00:00:00
Chapter 4 Building and running NGT Recover jobs 355
BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2015
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40700I EXECUTION PLAN SUMMARY:

BMC40701I PHASE: LOG INPUT
BMC40721I PHASE INITIALIZATION:
BMC40738I INITIALIZE LOG RECORD SORT 1
BMC40814I ESTIMATED RECORD SIZE = 1000
BMC40813I ESTIMATED NUMBER OF RECORDS = 797
BMC40738I ALLOCATE 2 POINTER BUFFERS (53092 BYTES EACH)
BMC40702I STEP: READ LOG PAGES
BMC40736I STEP INITIALIZATION:
BMC40738I ALLOCATE INPUT LOG FILE
BMC40302I DSN = DEFQCAT.LOGCOPY1.LOG11.DATA
BMC40741I END EXECUTION PLAN
FILE OPTIMIZATION
OPEN LOG DATASET
DSN = DEFQCAT.LOGCOPY1.DS11.DATA
ALLOCATE LOG DATASET BUFFER(S)
PREPARE LOG FOR PROCESSING
DSN = DEFQCAT.LOGCOPY1.DS11.DATA
ALLOCATE LOG DATASET BUFFER(S)
PREPARE LOG FOR PROCESSING
DSN = DEFQCAT.LOGCOPY1.DS11.DATA
CLOSE LOG DATASET
DEALLOCATE INPUT LOG FILE
DSN = DEFQCAT.LOGCOPY1.DS11.DATA
FREE 3 LOG DATASET BUFFER(S)
STEP TERMINATION:
END LOG RECORD SORT INPUT 1
FREE POINTER BUFFER
PHASE: MERGE
TABLESPACE = BMCDBSMP.BMCTS014
ALLOCATE 3 LOG RECORD BUFFER(S) (81920 BYTES EACH)
ALLOCATE INPUT IMAGE COPY FILE
FILE OPTIMIZATION
ALLOCATE 2 INPUT FILE BUFFER(S)
OPEN IMAGE COPY FILE
DSN = AFR.BMCDBSMP.BMCTS014.IC.G0003V00
ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
ALLOCATE SPACEMAP BUFFER (4096 BYTES)
ALLOCATE PASTE BUFFER (4096 BYTES)
SET RECOVERY PENDING (RECP) FLAG
CREATE SPACE
SPACE = BMCDBSMP.BMCTS014
ALLOCATE SPACE (READ/WRITE)
SPACE = BMCDBSMP.BMCTS014
CONNECT TO SPACE
ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
FILE OPTIMIZATION
ALLOCATE 2 SPACE OUTPUT BUFFER(S)
ALLOCATE SPACEMAP BUFFER (4096 BYTES)
ALLOCATE PASTE BUFFER (4096 BYTES)
STEP: MERGE SPACE
MERGE SPACE
TABLESPACE = BMCDBSMP.BMCTS014
PHASE TERMINATION:
DISCONNECT FROM SPACE
DEALLOCATE SPACE
SPACE = BMCDBSMP.BMCTS014
FREE PAGE CHECK BUFFER
FREE SPACEMAP BUFFER
FREE PASTE BUFFER
FREE 2 OUTPUT SPACE BUFFER(S)
FREE PAGE CHECK BUFFER
FREE PASTE BUFFER
FREE SPACEMAP BUFFER
CLOSE IMAGE COPY FILE
DSN = AFR.BMCDBSMP.BMCTS014.IC.G0003V00
FREE 2 INPUT FILE BUFFER(S)
DEALLOCATE INPUT IMAGE COPY FILE
DSN = AFR.BMCDBSMP.BMCTS014.IC.G0003V00
RESET RECOVERY PENDING (RECP) FLAG
END LOG RECORD SORT
FREE 3 LOG RECORD BUFFER(S)
REGISTER RECOVER TO CURRENT IN SYSCOPY
RECORD RESTART INFORMATION
PHASE: UNLOAD
### Examples of NGT Recover operations

This topic describes and provides examples of some of the recovery scenarios that you can implement by using NGT Recover.

The recovery scenarios show the different NGT Recover execution phases. NGT Recover analyzes all requests in the SYSIN data set and then constructs a plan that consists of the execution phases needed to accomplish the work requested. The following table describes the major execution phases.

<table>
<thead>
<tr>
<th>Phase Initialization</th>
<th>Phase Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TABLESPACE</strong> = BMCDBSMP.BMCTS014</td>
<td><strong>END KEY SORT INPUT FOR INDEX GROUP 1</strong></td>
</tr>
<tr>
<td><strong>INDEX</strong> = RMD.BMCIX014</td>
<td><strong>FREE 2 KEY BUFFER(S) FOR INDEX GROUP 1</strong></td>
</tr>
<tr>
<td><strong>ALLOCATE (2) KEY BUFFER(S) (270336 BYTES EACH) FOR INDEX GROUP 1</strong></td>
<td><strong>DISCONNECT FROM SPACE</strong></td>
</tr>
<tr>
<td><strong>INITIALIZE KEY SORT INPUT PHASE FOR INDEX GROUP 1</strong></td>
<td><strong>DEALLOCATE SPACE</strong></td>
</tr>
<tr>
<td><strong>ESTIMATED NUMBER OF RECORDS = 20012</strong></td>
<td><strong>SPACE = BMCDBSMP.BMCTS014</strong></td>
</tr>
<tr>
<td><strong>ALLOCATE SPACE (READ ONLY)</strong></td>
<td><strong>FREE SPACEMAP BUFFER</strong></td>
</tr>
<tr>
<td><strong>CONNECT TO SPACE</strong></td>
<td><strong>FREE PASTE BUFFER</strong></td>
</tr>
<tr>
<td><strong>FILE OPTIMIZATION</strong></td>
<td><strong>FREE PAGE CHECK BUFFER</strong></td>
</tr>
<tr>
<td><strong>ALLOCATE 2 SPACE INPUT BUFFER(S)</strong></td>
<td><strong>FREE 2 INPUT SPACE BUFFER(S)</strong></td>
</tr>
<tr>
<td><strong>ALLOCATE SPACEMAP BUFFER (4096 BYTES)</strong></td>
<td><strong>REGISTER REBUILD INDEX</strong></td>
</tr>
<tr>
<td><strong>ALLOCATE PASTE BUFFER (4096 BYTES)</strong></td>
<td><strong>RESET REALTIME STATISTICS</strong></td>
</tr>
<tr>
<td><strong>ALLOCATE PAGE CHECK BUFFER (4096 BYTES)</strong></td>
<td><strong>RECORD RESTART INFORMATION</strong></td>
</tr>
<tr>
<td><strong>UNLOAD KEYS</strong></td>
<td><strong>RESET REBUILD PENDING (RBDP) FLAG</strong></td>
</tr>
<tr>
<td><strong>BUILD INDEX</strong></td>
<td><strong>BUILD INDEX</strong></td>
</tr>
<tr>
<td><strong>FREE 2 SORT OUTPUT BUFFER(S) FOR INDEX GROUP 1</strong></td>
<td><strong>END SORT</strong></td>
</tr>
<tr>
<td><strong>DISCONNECT FROM SPACE</strong></td>
<td><strong>FREE 2 SORT OUTPUT BUFFER(S) FOR INDEX GROUP 1</strong></td>
</tr>
<tr>
<td><strong>DEALLOCATE SPACE</strong></td>
<td><strong>DISCONNECT FROM SPACE</strong></td>
</tr>
<tr>
<td><strong>SPACE = BMCDBSMP.BMCTS014</strong></td>
<td><strong>DEALLOCATE SPACE</strong></td>
</tr>
<tr>
<td><strong>REGISTER REBUILD INDEX</strong></td>
<td><strong>SPACE = BMCDBSMP.BMCTS014</strong></td>
</tr>
<tr>
<td><strong>RESET REALTIME STATISTICS</strong></td>
<td><strong>REGISTER REBUILD INDEX</strong></td>
</tr>
<tr>
<td><strong>RECORD RESTART INFORMATION</strong></td>
<td><strong>RESET REBUILD PENDING (RBDP) FLAG</strong></td>
</tr>
</tbody>
</table>
Table 12: Major NGT Recover execution phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG INPUT</td>
<td>The LOG INPUT phase reads the DB2 log records required for recovery and sends them to a sort task. If you specified MAXLOGS &gt; 1, NGT Recover starts multiple subtasks to handle log file allocation. The main task processes log records. The subtasks handle allocation.</td>
</tr>
<tr>
<td>MERGE</td>
<td>The MERGE phase reads sorted log records and applies them to page images, from an image copy or, in the case of BACKOUT and LOGONLY, from the space itself. This phase recovers table spaces or indexes from logs, extracts keys and sends them to a sort, reads the space for BACKOUT or LOGONLY recovery, writes output image copy data sets, writes output change accumulation files, and performs many other tasks. Multiple MERGE phases can run in parallel (using the MAXLSORT option).</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>The UNLOAD phase extracts the index keys from a table space or partition that is not being recovered (and has no MERGE phase) and sends those keys to a sort task. A MERGE phase and an UNLOAD phase should not occur for the same table space or table space partition.</td>
</tr>
<tr>
<td>BUILD</td>
<td>The BUILD phase constructs indexes by using sorted keys extracted in a MERGE or UNLOAD phase.</td>
</tr>
<tr>
<td>SNAP</td>
<td>The SNAP phase uses XBM or SUF to perform a data set snap to restore an Instant Snapshot copy.</td>
</tr>
</tbody>
</table>

Multiple table space recovery and index rebuilds

The following figure shows the phases that are used when multiple table spaces are being recovered and the associated indexes are being rebuilt and output copies are requested.

In this example, MAXLSORT is set to 1 so that multiple log sorts and parallel MERGE phases are not used. Log records are selected and sorted for all table spaces during the LOG INPUT phase and then are merged with the image copies of the table spaces during each MERGE phase. Index keys are also extracted during the MERGE phase and are passed directly to the sort routine. When the MERGE phase for a table space is complete, the BUILD phase constructs the indexes by using the sorted keys. The MERGE and BUILD phases repeat for each table space until all table spaces are recovered and all indexes are rebuilt. Sort tasks for log records and index
keys are managed by the appropriate phases and depend on the options selected. Sort tasks are shown as shaded boxes in the background in the following figure.

**Figure 66: Phases used for table space recovery with indexes rebuilt**

```sql
RECOVER TABLESPACE tableSpaceName1 OUTCOPY YES . . .

TABLESPACE tableSpaceName1
REBUILD INDEX(ALL) TABLESPACE tableSpaceName1 NOWORKDDN

REBUILD INDEX(ALL) TABLESPACE tableSpaceName1 NOWORKDDN
```
Parallel index key sorts and multitasking index rebuilds

The following figure shows the phases that are used when you use the MAXKSORT option to specify that index key sorts are to run in parallel and that index rebuilds are multitasked.

In this example, MAXLSORT is set to 1 so that multiple log sorts and parallel MERGE phases are not used and KSORTSHARE has no effect.

For a partitioned space, NGT Recover can schedule a BUILD phase for each partition of a partitioning index after the completion of the MERGE phase for the corresponding table space partition. In this case, the BUILD phase for the index partition can overlap the MERGE phase for the next partition. The BUILD phase for the partitions of a partitioned index can also be scheduled after all MERGE phases for the table space partitions have completed.

NGT Recover chooses the most efficient strategy to reduce sort times and the number of bytes that are sorted. The number of BUILD phases and index key sorts that can run concurrently is equal to the value of MAXKSORT.
Sort tasks are shown as shaded boxes in the background in the following figure.

**Figure 67: Phases used for table space recovery and parallel index sorts with multitasking index rebuilds**

Multiple log sorts, MERGE phases, and parallel index key sorts with multitasking index rebuilds

The following figure shows the phases that are used when you use:

- The MAXLSORT option to specify multiple log sorts and parallel MERGE phases
- The KSORTSHARE option to indicate that each MERGE phase has its own set of key sorts
- The MAXKSORT option to specify that index key sorts are to run in parallel and that index rebuilds are multitasked

NGT Recover schedules concurrent log sorts and MERGE phases for the table spaces. The indexes for each table space are handled as in “Parallel index key sorts”
and multitasking index rebuilds” on page 361, except that the total number of 
BUILD phases and index key sorts that can run concurrently is equal to the value of 
MAXKSORT * MAXLSORT.

NGT Recover chooses the most efficient strategy to reduce sort times and the 
number of bytes that are sorted.

Sort tasks are shown as shaded boxes in the background.

Figure 68: Phases used for table space recovery with multiple log sorts and parallel index sorts with 
multitasking index rebuilds

OPTIONS MAXKSORT 2 MAXLSORT 2 KSORTSHARE NO 
RECOVER TABLESPACE tableSpaceName1
      ...... 
      TABLESPACE tableSpaceName1 
      REBUILD INDEX(ALL) TABLESPACE tableSpaceName1 
      REBUILD INDEX(ALL) TABLESPACE tableSpaceName1
Multiple table space and index recoveries

The following figure shows the phases that are used when table spaces and index spaces are being recovered.

Log records are selected and sorted during the LOG INPUT phase and then are merged with the image copies of the spaces during the MERGE phases. The MERGE phase repeats for each table space and index. The sort task is shown as a shaded box in the background in the following figure.

Figure 69: Phases used for table space and index space recovery

RECOVER TABLESPACE tableSpaceName1
.....
TABLESPACE tableSpaceNameN
RECOVER INDEXSPACE(ALL) TABLESPACE tableSpaceName1
.....
RECOVER INDEXSPACE(ALL) TABLESPACE tableSpaceNameN

BACKOUT recovery

The following figure shows the phases that are used when a table space and an index space are being recovered with the BACKOUT option to a specific point in time. Log records are selected and sorted during the LOG INPUT phase and then are merged with the table space and the index space. Image copies are not used with the BACKOUT strategy. In this case only active log records are needed. (Log records since the last quiesce are all that are required.)

Instant Snapshot recovery

The following figure shows the phases that are used when table spaces and index spaces are being recovered by using an Instant Snapshot copy.

If the recovery is a TOCOPY request or if there is no log to apply and no output copies, the SNAP phase does not require the following LOG INPUT phase or MERGE phase. A MERGE phase is scheduled if keys are being extracted or output copies are being made.
Examples of NGT Recover operations

Figure 71: Phases used for table space and index space recovery for Instant Snapshots

![Diagram showing phases of recovery process]

- **SNAP PHASE**: Instant Snapshot copy
- **LOG INPUT PHASE**: Archive log, Active log
- **MERGE PHASE**: Log sort task, Table space, Index space

Options for recovering table spaces and index spaces:

- OPTIONS XBMID DBXM
- RECOVER TABLESPACE tableSpaceName1
- TABLESPACE tableSpaceName
  RECOVER IDXSPACE(ALL) TABLESPACE tableSpaceName1
- RECOVER IDXSPACE(ALL) TABLESPACE tableSpaceName
Examples of NGT Recover jobs

This chapter provides example JCL for commonly-run NGT Recover jobs. Example output is provided for some of the examples. For more information on the output produced by NGT Recover jobs, see the following topics:

- “NGT Recover data sets and NGT Recover DD statements” on page 319
- “DD statements common to all NGT Recover executions” on page 319
- “Using MSGLEVEL to control NGT Recover output” on page 341

See “NGT Recover syntax” on page 83 for a description of the syntax options used in these examples.

Copies of the JCL for these examples are in members AFREXnn (where nn is the example number) in the HLQ.AFRSAMP installation data set (where HLQ represents the high-level qualifier specified during installation).

The following tables provide cross references based on the utility parameters, commands, or keywords used in the examples. This table gives the cross references for the utility parameters first, followed by the commands, in the order they are most likely to be used, and their keywords in SYSIN.

### Table 13: Job examples for Utility parameters

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARM VALUE</td>
<td>All examples</td>
</tr>
<tr>
<td>REGION</td>
<td>All examples</td>
</tr>
<tr>
<td>Command or keyword</td>
<td>Relevant examples</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td><strong>NEW/RESTART</strong></td>
<td>“Example 1: Previewing the recovery plan” on page 373</td>
</tr>
<tr>
<td></td>
<td>“Example 2B: Using RECOVER INDEX” on page 383</td>
</tr>
<tr>
<td></td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397 through “Example 9: Recovering to a non-DB2 data set” on page 404</td>
</tr>
<tr>
<td></td>
<td>“Example 10B: Overriding the IXRECP installation option” on page 416</td>
</tr>
<tr>
<td></td>
<td>“Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
<tr>
<td></td>
<td>“Example 13: Using the LOGSCAN command” on page 423 through “Example 17: Using timestamp recovery” on page 435</td>
</tr>
<tr>
<td><strong>NEW</strong></td>
<td>“Example 2A: Using REBUILD INDEX” on page 378</td>
</tr>
<tr>
<td></td>
<td>“Example 10A: Overriding several installation options” on page 415</td>
</tr>
<tr>
<td><strong>MSGLEVEL</strong></td>
<td>“Example 2: Recovering to the current state” on page 378 through “Example 9: Recovering to a non-DB2 data set” on page 404</td>
</tr>
<tr>
<td></td>
<td>“Example 10B: Overriding the IXRECP installation option” on page 416 through “Example 17: Using timestamp recovery” on page 435</td>
</tr>
<tr>
<td><strong>MAINT</strong></td>
<td>“Example 12: Using the MAINT parameter” on page 423</td>
</tr>
</tbody>
</table>

Table 14: Job examples using the OPTIONS command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OPTIONS</strong></td>
<td>“Example 2: Recovering to the current state” on page 378</td>
</tr>
<tr>
<td></td>
<td>“Example 5: Recovering to a specific log point” on page 400</td>
</tr>
<tr>
<td><strong>BACKOUT</strong></td>
<td>“Example 17: Using timestamp recovery” on page 435</td>
</tr>
<tr>
<td><strong>EARLYRECALL</strong></td>
<td>“Example 2: Recovering to the current state” on page 378</td>
</tr>
<tr>
<td><strong>INDEXLOG</strong></td>
<td>“Example 2: Recovering to the current state” on page 378</td>
</tr>
<tr>
<td></td>
<td>“Example 5: Recovering to a specific log point” on page 400</td>
</tr>
<tr>
<td></td>
<td>“Example 13: Using the LOGSCAN command” on page 423</td>
</tr>
<tr>
<td><strong>IXRECP</strong></td>
<td>“Example 10: Overriding installation options” on page 415</td>
</tr>
</tbody>
</table>
### Table 15: Job examples using the OUTPUT command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 17: Using timestamp recovery” on page 435</td>
</tr>
<tr>
<td>DSNNAME</td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 17: Using timestamp recovery” on page 435</td>
</tr>
<tr>
<td>MODELDCB</td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397</td>
</tr>
<tr>
<td>SPACE</td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397</td>
</tr>
<tr>
<td>UNIT</td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 17: Using timestamp recovery” on page 435</td>
</tr>
</tbody>
</table>
### Table 16: Job examples using the RECOVER command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER</td>
<td>“Example 1: Previewing the recovery plan” on page 373 through “Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
<tr>
<td></td>
<td>“Example 8: Recovering using tape-stacked data sets” on page 403 through “Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
<tr>
<td></td>
<td>“Example 16: Using MAXKSORT and recovering a table space” on page 434 through “Example 17: Using timestamp recovery” on page 435</td>
</tr>
<tr>
<td>ANALYZE</td>
<td>“Example 1: Previewing the recovery plan” on page 373</td>
</tr>
<tr>
<td>BACKOUT</td>
<td>“Example 5: Recovering to a specific log point” on page 400</td>
</tr>
<tr>
<td>DSNUM</td>
<td>“Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
<tr>
<td>INDEP OUTSPACE</td>
<td>“Example 9: Recovering to a non-DB2 data set” on page 404</td>
</tr>
<tr>
<td>INDEX</td>
<td>“Example 5: Recovering to a specific log point” on page 400</td>
</tr>
<tr>
<td></td>
<td>“Example 17: Using timestamp recovery” on page 435</td>
</tr>
<tr>
<td>LASTCOPY</td>
<td>“Example 4: Recovering to a specified copy” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 8: Recovering using tape-stacked data sets” on page 403</td>
</tr>
<tr>
<td></td>
<td>“Example 10: Overriding installation options” on page 415</td>
</tr>
<tr>
<td>LASTQUIESCE</td>
<td>“Example 5: Recovering to a specific log point” on page 400</td>
</tr>
<tr>
<td>OUTCOPY</td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397</td>
</tr>
<tr>
<td>OUTCOPYDDN</td>
<td>“Example 3: Creating and dynamically allocating output copies” on page 397</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>“Example 1: Previewing the recovery plan” on page 373 through “Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
<tr>
<td></td>
<td>“Example 8: Recovering using tape-stacked data sets” on page 403 through “Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
<tr>
<td></td>
<td>“Example 16: Using MAXKSORT and recovering a table space” on page 434</td>
</tr>
<tr>
<td>TOCOPY</td>
<td>“Example 4: Recovering to a specified copy” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 8: Recovering using tape-stacked data sets” on page 403</td>
</tr>
<tr>
<td></td>
<td>“Example 10: Overriding installation options” on page 415</td>
</tr>
<tr>
<td>TOLOGPOINT</td>
<td>“Example 5: Recovering to a specific log point” on page 400</td>
</tr>
</tbody>
</table>
### Table 17: Job examples using the REBUILD command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>REBUILD</td>
<td>“Example 2: Recovering to the current state” on page 378</td>
</tr>
<tr>
<td></td>
<td>“Example 4: Recovering to a specified copy” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
<tr>
<td></td>
<td>“Example 9: Recovering to a non-DB2 data set” on page 404</td>
</tr>
<tr>
<td></td>
<td>“Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
<tr>
<td></td>
<td>“Example 15: Using MAXKSORT for parallel index rebuilds” on page 426</td>
</tr>
<tr>
<td></td>
<td>“Example 16: Using MAXKSORT and recovering a table space” on page 434</td>
</tr>
<tr>
<td>INDEP OUTSPACE</td>
<td>“Example 9: Recovering to a non-DB2 data set” on page 404</td>
</tr>
<tr>
<td>INDEX</td>
<td>“Example 2: Recovering to the current state” on page 378</td>
</tr>
<tr>
<td></td>
<td>“Example 4: Recovering to a specified copy” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
<tr>
<td></td>
<td>“Example 9: Recovering to a non-DB2 data set” on page 404</td>
</tr>
<tr>
<td></td>
<td>“Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
<tr>
<td></td>
<td>“Example 15: Using MAXKSORT for parallel index rebuilds” on page 426</td>
</tr>
<tr>
<td></td>
<td>“Example 16: Using MAXKSORT and recovering a table space” on page 434</td>
</tr>
<tr>
<td>NOWORKDDN</td>
<td>“Example 2: Recovering to the current state” on page 378</td>
</tr>
<tr>
<td></td>
<td>“Example 4: Recovering to a specified copy” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 9: Recovering to a non-DB2 data set” on page 404</td>
</tr>
<tr>
<td></td>
<td>“Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
<tr>
<td>PART</td>
<td>“Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>“Example 4: Recovering to a specified copy” on page 397</td>
</tr>
<tr>
<td></td>
<td>“Example 15: Using MAXKSORT for parallel index rebuilds” on page 426</td>
</tr>
<tr>
<td></td>
<td>“Example 16: Using MAXKSORT and recovering a table space” on page 434</td>
</tr>
</tbody>
</table>
### Table 18: Job examples using the RECOVER UNLOADKEYS command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER UNLOADKEYS</td>
<td>“Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
<tr>
<td>PART</td>
<td>“Example 6: Extracting nonpartitioned index keys” on page 402</td>
</tr>
</tbody>
</table>

### Table 19: Job examples using the RECOVER BUILDINDEX command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOVER BUILDINDEX</td>
<td>“Example 7: Building a nonpartitioned index” on page 402</td>
</tr>
</tbody>
</table>

### Table 20: Job examples using the ACCUM command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCUM</td>
<td>“Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
<tr>
<td>GROUP</td>
<td>“Example 11: Recovering using a change accumulation file” on page 422</td>
</tr>
</tbody>
</table>

### Table 21: Job examples using the LOGSCAN command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGSCAN</td>
<td>“Example 13: Using the LOGSCAN command” on page 423</td>
</tr>
<tr>
<td>BACKOUT</td>
<td>“Example 13: Using the LOGSCAN command” on page 423</td>
</tr>
<tr>
<td>INDEXSPACE</td>
<td>“Example 13: Using the LOGSCAN command” on page 423</td>
</tr>
<tr>
<td>LASTQUIESCE</td>
<td>“Example 13: Using the LOGSCAN command” on page 423</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>“Example 13: Using the LOGSCAN command” on page 423</td>
</tr>
<tr>
<td>TOLOGPOINT</td>
<td>“Example 13: Using the LOGSCAN command” on page 423</td>
</tr>
</tbody>
</table>

### Table 22: Job examples using the SIMRCVR command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMRCVR</td>
<td>“Example 14: Simulating recovery” on page 426</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>“Example 14: Simulating recovery” on page 426</td>
</tr>
</tbody>
</table>

### Table 23: Job examples using the SIMRBLD command

<table>
<thead>
<tr>
<th>Command or keyword</th>
<th>Relevant examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMRBLD</td>
<td>“Example 14: Simulating recovery” on page 426</td>
</tr>
</tbody>
</table>
Example 1: Previewing the recovery plan

This example illustrates the use of the ANALYZE ONLY option with RECOVER TABLESPACE to obtain a preview of the recovery plan.

When you use the ANALYZE ONLY option, NGT Recover determines the data sets required for recovery and produces a recovery plan. This example also uses the SYSPICK DD statement to list all of the tape and cartridge volumes that will be dynamically allocated during recovery. The JCL for this example is in the following figure.

Figure 72: Example 1—JCL using ANALYZE ONLY

```
//AFREX01 JOB (PAFR), 'EXAMPLE 1',
** CLASS=Q,
** MSGCLASS=X, NOTIFY=&SYSUID
/*ROUTE XEQ BMCPLX1
/*JOBPARM SYSAFF=DB2A
//*
//RECOVER EXEC PGM=AFRMAIN,REGION=0M,
//             PARM='DGE,EXAMPLE01,NEW/RESTART'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//         DD DISP=SHR, DSN=DB2.DSNEXIT
//         DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
RECOVER TABLESPACE BMCDBSMP.BMCTS001
   LOGSORT ANALYZE ONLY
/*
//SYSPICK DD SYSOUT=*n
```

Figure 73 on page 373 through Figure 77 on page 378 show the following output files from this NGT Recover job:

- AFRSUMRY
- AFRSTMT
- AFROSUM
- AFRPLAN
- SYSPICK

See Table 9 on page 320 for a description of each of these files.

Figure 73: Example 1—AFRSUMRY using ANALYZE ONLY
Figure 74: Example 1—AFRSTMT using ANALYZE ONLY

Example 1: Previewing the recovery plan
Example 1: Previewing the recovery plan

Chapter 5  Examples of NGT Recover jobs  375
Example 1: Previewsing the recovery plan

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>08/17/2016</td>
<td>12:58:11</td>
<td>ANALYZE STARTING</td>
</tr>
<tr>
<td>08/17/2016</td>
<td>12:58:11</td>
<td>CATALOG LOOKUP STARTING</td>
</tr>
<tr>
<td>08/17/2016</td>
<td>12:58:11</td>
<td>USING STANDARD DB2 SECURITY</td>
</tr>
<tr>
<td>08/17/2016</td>
<td>12:58:11</td>
<td>SYNCHRONIZATION STARTING</td>
</tr>
<tr>
<td>08/17/2016</td>
<td>12:58:11</td>
<td>SYNCHRONIZATION FINISHED</td>
</tr>
<tr>
<td>08/17/2016</td>
<td>12:58:11</td>
<td>PLANNING STARTING</td>
</tr>
<tr>
<td>08/17/2016</td>
<td>12:58:11</td>
<td>TIME TO READ SYSLGRNX WAS 00:00:00</td>
</tr>
</tbody>
</table>

Figure 75: Example 1—AFROSUM using ANALYZE ONLY
### Example 1: Previewing the recovery plan

**Figure 76: Example 1—AFRPLAN using ANALYZE ONLY**

<table>
<thead>
<tr>
<th>BMC40018I</th>
<th>NGT RECOVER FOR DB2 V12.1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40473I</td>
<td>COPYRIGHT BMC SOFTWARE INC. 1991-2016</td>
</tr>
<tr>
<td>BMC96173I</td>
<td>RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001</td>
</tr>
<tr>
<td>BMC40474I</td>
<td>CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO <a href="mailto:SUPPORT@BMC.COM">SUPPORT@BMC.COM</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BMC40755I</th>
<th>ANALYSIS PLAN SUMMARY:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40716I</td>
<td>STOGROUP = SYSDEFLT PRIQTY = 800 SECQTY = 400</td>
</tr>
<tr>
<td>BMC40710I</td>
<td>PAGE SIZE = 4K SEGSIZE = 4</td>
</tr>
<tr>
<td>BMC40723I</td>
<td>ENCODING SCHEME = EBCDIC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BMC40820I</th>
<th>TABLESPACE RECOVERY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40827I</td>
<td>LOG RESOURCE RANGE:</td>
</tr>
<tr>
<td>BMC40585I</td>
<td>FROM LOGPOINT = X'00CA088B59E63E000000' TO LOGPOINT = X'00CA08A2FF9049000000'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BMC40840I</th>
<th>INPUT COPY / SYSCOPY DATA:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40640I</td>
<td>2012-08-17 11.12.24.2272 RBA/LRSN X'00CA088B59E63E000000' FULL IMAGE COPY</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = RDAPXR.BMCDBSMP.BMCTS001.F.T111223</td>
</tr>
<tr>
<td>BMC40641I</td>
<td>SHRLEVEL = REFERENCE SITETYPE = LP DSNUM = 0</td>
</tr>
<tr>
<td>BMC40640I</td>
<td>2012-08-17 11.12.26.9237 RBA/LRSN X'00CA088B5D166C000000' QUIESCE UTILITY POINT - WRITE(YES)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BMC40716I</th>
<th>PHASE: MERGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40721I</td>
<td>PHASE INITIALIZATION:</td>
</tr>
<tr>
<td>BMC40301I</td>
<td>TABLESPACE = BMCDBSMP.BMCTS001</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>ALLOCATE INPUT IMAGE COPY FILE</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = RDAPXR.BMCDBSMP.BMCTS001.F.T111223</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>FILE OPTIMIZATION</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>ALLOCATE 2 INPUT FILE BUFFER(S)</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>OPEN IMAGE COPY FILE</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = RDAPXR.BMCDBSMP.BMCTS001.F.T111223</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>SET RECOVERY PENDING (RECP) FLAG</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>CREATE SPACE</td>
</tr>
<tr>
<td>BMC40345I</td>
<td>SPACE = BMCDBSMP.BMCTS001</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>ALLOCATE SPACE (READ/WRITE)</td>
</tr>
<tr>
<td>BMC40345I</td>
<td>SPACE = BMCDBSMP.BMCTS001</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>CONNECT TO SPACE</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>FILE OPTIMIZATION</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>ALLOCATE 2 SPACE OUTPUT BUFFER(S)</td>
</tr>
<tr>
<td>BMC40702I</td>
<td>STEP: MERGE SPACE</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>MERGE SPACE</td>
</tr>
<tr>
<td>BMC40301I</td>
<td>TABLESPACE = BMCDBSMP.BMCTS001</td>
</tr>
<tr>
<td>BMC40735I</td>
<td>PHASE TERMINATION:</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>DISCONNECT FROM SPACE</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>DEALLOCATE SPACE</td>
</tr>
<tr>
<td>BMC40345I</td>
<td>SPACE = BMCDBSMP.BMCTS001</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>FREE 2 OUTPUT SPACE BUFFER(S)</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>CLOSE IMAGE COPY FILE</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = RDAPXR.BMCDBSMP.BMCTS001.F.T111223</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>FREE 2 INPUT FILE BUFFER(S)</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>DEALLOCATE INPUT IMAGE COPY FILE</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = RDAPXR.BMCDBSMP.BMCTS001.F.T111223</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>RESET RECOVERY PENDING (RECP) FLAG</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>REGISTER RECOVER TO CURRENT IN SYSCOPY</td>
</tr>
<tr>
<td>BMC40738I</td>
<td>RECORD RESTART INFORMATION</td>
</tr>
</tbody>
</table>
Example 2: Recovering to the current state

This example illustrates two ways of recovering a partitioned table space and an index on the table in the table space to the current state.

One job uses REBUILD INDEX (“Example 2A: Using REBUILD INDEX” on page 378), and the other job uses RECOVER INDEX (“Example 2B: Using RECOVER INDEX” on page 383). The EARLYRECALL option is also used in both jobs, causing data sets to be scheduled for recall before other activities begin.

Example 2A: Using REBUILD INDEX

In the first job in this example, the index is rebuilt from the rows in this recovered image.

Note the use of the NOWORKDDN option to send the extracted index keys directly to the sort routine. Also, note the use of MSGLEVEL(0), which produces less output than the default of MSGLEVEL(1).

Figure 78: Example 2A—JCL for a simple RECOVER TABLESPACE and REBUILD INDEX

```plaintext
//AFREX02A JOB (5225), 'EXAMPLE 2A',
//   CLASS=Q,
//   MSGCLASS=X, NOTIFY=&SYSUID
/*ROUTE XEQ BMCPLX1
/*JOBPARM SYSAFF=DB2A
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=OM,
//   PARM='DGE,EX02AMSG0,NEW,MSGLEVEL(0)'
//STEPLIB DD DISP=SHR, DSN=product.libraries
// DD DISP=SHR, DSN=DB2.DSNEXIT
// DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
// OPTIONS EARLYRECALL INDEXLOG NO
RECOVER TABLESPACE BMCDBSMP.BMCDBSMP.
REBUILD INDEX (RMD.BMCIX011) NOWORKDDN
/*
//
```
Figure 79 on page 379 through Figure 81 on page 382 show the output produced by this JCL, which includes the following files:

- AFRSUMRY
- AFRSTMT
- AFRPR001

For a description of each of these files, see Table 9 on page 320.

Figure 79: Example 2A—AFRSUMRY for a simple REBUILD INDEX

```plaintext
BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM
BMC40001I UTILITY EXECUTION STARTING 08/17/2012 12:59:30
BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40876I MAINT:  NO RECOVER PLUS PTFS APPLIED
BMC96010I SOLUTION COMMON CODE V11.1.01
BMC40876I MAINT:  NO SOLUTION COMMON CODE PTFS APPLIED
BMC96010I BMCSORT ENGINE V2.3.01
BMC40876I MAIN:  BPJ0195 BPJ0198 BPJ0209 BPJ0250 BPJ0263 BPJ0269 BPJ0279
BMC40876I MAINT:  BPJ0288 BPJ0308 BPJ0343 BPJ0357 BPJ0361 BPJ0374 BPJ0390
BMC40876I MAINT:  BPJ0404 BPJ0473 BPJ0503 BPJ0521
BMC40002I UTILITY ID = EX02AMSG0. DB2 SUBSYSTEM ID = DXW .
BMC40094I ATTEMPTING TO CONNECT TO DB2 SUBSYSTEM DXW , USING PLAN AFRB111T
BMC40342I A GROUP ATTACH WAS PERFORMED WITH DXW
BMC40024I SUCCESSFUL CONNECT TO W4 (RELEASE 910 OF DB2) USING PLAN AFRB111T
BMC40067I DB2 W4 IS MEMBER DXW4 IN DATA SHARING GROUP DSNDXW
BMC40475I EXECUTION SUMMARY
BMC40477I      TABLE SPACES / PHASES (PARTITIONS) RECOVERED   :  1 (0)
BMC40479I      LOG RECORDS APPLIED                   :  22496
BMC40480I      INDEXES REBUILT                       :  1
BMC40005I UTILITY EXECUTION COMPLETE 08/17/2012 12:59:41, RETURN CODE = 0
```

Figure 80: Example 2A—AFRSTMT for a simple REBUILD INDEX

```plaintext
BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40336I ANALYZE FINISHED.
BMC40287I     ELAPSED TIME = 00:00:03, UTILITY ELAPSED TIME = 00:00:04
BMC40288I     CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00
BMC40336I PLANNING FINISHED.
BMC40287I     ELAPSED TIME = 00:00:01, UTILITY ELAPSED TIME = 00:00:05
BMC40288I     CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00
BMC40336I PLAN EXECUTION FINISHED.
BMC40287I     ELAPSED TIME = 00:00:05, UTILITY ELAPSED TIME = 00:00:10
BMC40288I     CPU TIME     = 00:00:00, UTILITY CPU TIME     = 00:00:00
BMC40476I EXECUTION SUMMARY
BMC40477I     TABLE SPACES / PHASES (PARTITIONS) RECOVERED    :  1 (0)
BMC40479I     LOG RECORDS APPLIED                          :  22496
BMC40480I     INDEXES REBUILT                             :  1
BMC40005I UTILITY EXECUTION COMPLETE 08/17/2012 12:59:41, RETURN CODE = 0
```
Example 2: Recovering to the current state

BMC40473I COPYRIGH BMC SOFTWARE INC. 1991-2013
BMC406173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC4074I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40931I PARM LIST:   DB2 SSID = DXW
BMC40933I                UTILID = EX02AMSG0
BMC40930I UTILITY REPLACEMENT ('NEW') REQUESTED
BMC40934I               RESTART = NEW
BMC40930I              MSGLEVEL = 0
BMC40930I        PARM NOT CODED - CHECKPT
BMC40930I        PARM NOT CODED - RDB2STAT
BMC40930I        PARM NOT CODED - AFROPTS
BMC40930I        PARM NOT CODED - ACAOPTS
BMC40935I ----- DEFAULT OPTIONS:
BMC40924I              PLANRECV = AFRB111T       (MAIN PLAN)
BMC40408I         BINDQUALIFIER = BMCAFR         (QUALIFIER USED FOR DYNAMIC BIND)
BMC40241I            PUBLICPLAN = YES            (GRANT PLAN TO PUBLIC)
BMC96003I                  ZIIP = ENABLED        (ZIIP PROCESSOR ENABLEMENT)
BMC40941I                IXRECP = NO             (SET INDEX RECOVER PENDING)
BMC40878I                  WTOR = YES            (ISSUE WTOR IF SPACE REMAINS IN STOPP
BMC40934I                WKUNIT = SYSALLDA       (WORK UNIT NAME)
BMC40927I              RDB2STAT = YES            (RESET DB2 OBJECT STATUS)
BMC40946I              OPNDB2ID = YES            (ACQUIRE RACF AUTHORITY OF DB2)
BMC40938I                AMSCAT = NO             (USE 'CATALOG' IN IDCAMS INPUT)
BMC40928I                SMCORE = (    0K,   0K) (SORT CORE VALUES)
BMC40934I USEHDROBIDS = YES (USE HEADER PAGE IDS FOR OBIDXLAT)
BMC96235I               SORTDEVT = SORT DEFAULT   (DEVICE TYPE FOR LOG AND INDEX SORTS)
BMC40962I              SORTNUM = SORT DEFAULT (WORK DATASETS FOR LOG AND INDEX SORTS)
BMC96230I               KSORTSHARE = YES (KEY SortS SHARED BY ALL TASKS)
BMC40942I                ACCUMS = (LP,LB)        (DEFAULT SEQUENCE FOR CA FILES)
BMC96112I                  LOGS = (ACT1,ACT2     (DEFAULT SEQUENCE FOR LOG FILES)
BMC96112I                          ARC1,ARC2)
BMC96035I ----- DEFAULT OPTIONS:  SORT
BMC40964I                EATTR = NONE           (EXTENDED ATTRIBUTES FOR DYNAMIC
BMC40925I              MAXKSORT = 10  (DEFAULT)  (MAX CONCURRENT INDEX KEY SORTS)
BMC40891I               MAXLOGS = 3              (DEFAULT MAX CONCURRENT LOG FILES)
BMC96035I ----- DEFAULT OPTIONS:  OUTPUT COPIES
BMC40962I                 OUTCOPY = ASCODED (OUTCOPY CREATION DIRECTIVE)
BMC96221I               AUTOsize = YES       (AUTO SIZE OUTPUT COPIES AND OUTPUT
BMC96033I                EATTR = NONE           (EXTENDED ATTRIBUTES FOR DYNAMIC
BMC96035I ----- DEFAULT OPTION: ALLOCATION OF OUTPUT COPIES:
BMC96035I ----- DEFAULT OPTIONS:  ESTIMATION
BMC40961I                COPIES = (FC,LP,LB) (DEFAULT SEQUENCE FOR IMAGE COPIES)
BMC96112I               LOGS = (ACT1,ACT2)  (DEFAULT SEQUENCE FOR LOG FILES)
BMC96112I ARCI,ARC2)
BMC96035I ----- DEFAULT RESOURCE SELECTION SEQUENCE FOR REMOTE SITE
BMC96112I               LOGS = (ACT1,ACT2)  (DEFAULT SEQUENCE FOR LOG FILES)

380  BMC Next Generation Technology Recover for DB2 for z/OS Reference Manual
Example 2: Recovering to the current state

Chapter 5  Examples of NGT Recover jobs  381
BMC40336I SYNCHRONIZATION FINISHED.
BMC40287I ELAPSED TIME = 00:00:02, UTILITY ELAPSED TIME = 00:00:04
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC96306I MAXLSORT REDUCED FROM 50 TO 17 DUE TO MEMORY CONSTRAINTS

BMC40335I PLANNING STARTING 08/17/2012 12:59:34

BMC40512I LOG FILE RESOURCES:
BMC40302I DSN = DSNDXW.DXW4.LOGCOPY1.DS01.DATA
BMC40566I USING LOG DSNDXW.DXW4.LOGCOPY1.DS01.DATA FOR PROCESSING
BMC40555I FROMRBA = X'00000000015B9138D000' TORBA = X'00000000015B914A5E0C'
BMC40555I FROMRBA = X'00000000015B914CD000' TORBA = X'00000000015B915A53E5'

BMC40335I PLAN EXECUTION STARTING 08/17/2012 12:59:36

Figure 81: Example 2A--AFRP001 for a simple REBUILD INDEX

BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40483I LOG INPUT COMPLETE.
BMC40483I MERGE COMPLETE.
BMC40301I TABLESPACE = BMCDBSMP.BMCTS011
BMC40483I BUILD COMPLETE.

BMC40971I 1 SYNCSORT FOR Z/OS 2.3.1.0NI U.S. PATENTS: 4210961, 5117495 (C) 2007 SYNCSORT INC.
BMC40483I DATE=2012/230 TIME=12.59.36
BMC40971I BMCSORT ENGINE - IF PROBLEMS, CONTACT BMC SUPPORT AT 1-800-537-1813

z/OS 1.13.0

$BMCPRM$ : BMSG,EQUALS
PARMLIST :
SORT FIELDS=(30,20,BI,A),DYNALLOC,FILSZ=E664
RECORD TYPE=V,LENGTH=(8192,8192,8192,32,1000)
OPTION SORTDD=L001,MSGDDN=LOGOU001,AVGRLEN=1000
WER161B ALTERNATE PARM USED
WER164B 2,260K BYTES OF VIRTUAL STORAGE AVAILABLE, MAX REQUESTED,
WER164B 44K BYTES RESERVE REQUESTED, 2,212K BYTES USED
WER410B 1,956K BYTES OF VIRTUAL STORAGE AVAILABLE ABOVE THE 16MEG LINE,
WER410B 0 BYTES RESERVE REQUESTED, 1,956K BYTES USED
WER036B G=150,B=13672,SEGLEN=4200,8192=34
WER162B 0 PREALLOCATED SORTWORK TRACKS, 960 DYNAMICALLY ALLOCATED,
WER162B 0 ACQUIRED IN SECONDARY EXTENTS, 0 RELEASED, TOTAL OF 40 TRACKS USED
WER045C END SORT PHASE
WER051I INSERT 22496, DELETE 22496
WER416B L001WK01 : EXCP'S=7,UNIT=3390,DEV=40AB,CHP=89DEEF,Vol=WRK029
WER416B L001WK02 : EXCP'S=6,3390,DEV=5949,CHP=FDE8E,Vol=WRK069
WER416B TOTAL OF 13 EXCP'S ISSUED FOR SORTWORKS
WER416B TOTAL OF 13 EXCP'S ISSUED FOR SORTING
WER246I FILESIZE 1,609,708 BYTES
WER054I RCD IN 0, OUT 0
WER072I EQUALS, ELAP IN EFFECT
WER169I RELEASE 2.3 BATCH 0486 TPF LEVEL 0.0
WER052I END SYNCSORT - AFREX02A,RECOVER,,DIAG=E000,EADE,C83C,20C4,A9C2,4DC2,0260,0C60
Example 2B: Using RECOVER INDEX

In the second job in this example, the index is recovered from image copies and log.

Note that this job uses MSGLEVEL(2), which produces all of the output produced by the default of MSGLEVEL(1) and the AFRPLAN.

Figure 82: Example 2B—JCL for a simple RECOVER TABLESPACE and RECOVER INDEX

```plaintext
//AFREX02B JOB (PAFR),'EXAMPLE 2B',
//         CLASS=Q,
//         MSGCLASS=X,NOTIFY=&SYSUID
/*ROUTE XEQ BMCPLX1
/*JOBPARM SYSAFF=DB2A
//*
//RECOVER EXEC PGM=AFRMAIN,REGION=OM,
//             PARM='DGE,EX02B,NEW/RESTART,MSGLEVEL(2)'
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
OPTIONS EARLYRECALL INDEXLOG YES
RECOVER TABLESPACE BMCDBSMP.BMCTS011
RECOVER INDEX (RMD.BMCIX011)
/*
```

Figure 83 on page 383 through Figure 89 on page 394 show the following output files produced by this JCL:

- AFRSUMRY
- AFRSTMT
- AFRPRINT
- AFRPR001
- AFRPR002
- AFROSUM
- AFRPLAN

For a description of each of these files, see Table 9 on page 320.

Figure 83: Example 2B—AFRSUMRY for a simple RECOVER INDEX

```plaintext
BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC408741 CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM
BMC40001I UTILITY EXECUTION STARTING 08/17/2012 13:39:16
BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC408761 MAINT: NO RECOVER PLUS PTFS APPLIED
BMC96010I SOLUTION COMMON CODE V11.1.01
BMC408761 MAINT: NO SOLUTION COMMON CODE PTFS APPLIED
```
Figure 84: Example 2B—AFRSTMT for a simple RECOVER INDEX

BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40471I TABLE SPACES / PHASES (PARTITIONS) RECOVERED :  1 (0)
BMC40478I INDEX SPACES / PHASES (PARTITIONS) RECOVERED :  1 (0)
BMC40479I LOG RECORDS APPLIED                   :  12498

BMC40005I UTILITY EXECUTION COMPLETE 08/17/2012 13:39:36, RETURN CODE = 0
Example 2: Recovering to the current state

Chapter 5  Examples of NGT Recover jobs  385
Example 2: Recovering to the current state

BMC96010I SOLUTION COMMON CODE V11.1.01
BMC40876I MAINT: NO SOLUTION COMMON CODE PTFS APPLIED

BMC96010I BMCSORT ENGINE V2.3.01
BMC40876I MAINT: BPJ0195 BPJ0198 BPJ0209 BPJ0250 BPJ0263 BPJ0269 BPJ0279
BMC40876I MAINT: BPJ0288 BPJ0308 BPJ0343 BPJ0357 BPJ0361 BPJ0374 BPJ0390
BMC40876I MAINT: BPJ0404 BPJ0473 BPJ0503 BPJ0521

BMC40335I PACKAGE CHECKING AND BINDING STARTING 08/17/2012 13:39:17
BMC40336I PACKAGE CHECKING AND BINDING FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40020I BYT:0, 0, 0; BLK:0, 0, 0; TOT:860944, 5115972, 5115972; iter: 1; RC:4

BMC40611I INPUT STATEMENTS:
BMC40101I OPTION EARLYRECALL INDEXLOG YES
BMC40101I RECOVER TABLESPACE BMCDSM.BMCTS011
BMC40101I RECOVER INDEX (RMD.BMCIX011)
BMC40101I
BMC40959I INDEXLOG YES STRATEGY REQUESTED
BMC96020I USE OF REAL-TIME STATISTICS ENABLED FOR REBUILD SORT ESTIMATION
BMC96004I ZIIP ENABLED (0) USING SUBSYSTEM ACP1
BMC40937I z/OS RELEASE = 1.13.00, PID = HBB7780, SMS RELEASE = 1.13.0
BMC40157I AVAILABLE REGION BELOW 16M = 10056K, AVAILABLE REGION ABOVE 16M = 1362352K,
NUMBER OF CPUS = 5
BMC40037I UTILID EX02B WAS NOT FOUND IN THE BMCUTIL TABLE - UTILITY WILL RUN AS IF 'NEW' WERE SPECIFIED

BMC40335I ANALYZE STARTING 08/17/2012 13:39:17
BMC40335I CATALOG LOOKUP STARTING 08/17/2012 13:39:17
BMC40336I PROC RO STMT FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSTB FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSIX FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSKY FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I TS LOOKUP FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I USR LOG LOOUP FINISHED.
BMC40336I DB LOOKUP FINISHED.
Example 2: Recovering to the current state

Chapter 5  Examples of NGT Recover jobs  387
Example 2: Recovering to the current state

```
BMC40302I  DSN = DSNDXW.DXW1.LOGCOPY1.DS02.DATA
BMC40302I  DSN = DSNDXW.DXW3.LOGCOPY1.DS02.DATA
BMC40302I  DSN = DSNDXW.DXW4.LOGCOPY1.DS01.DATA
BMC40302I  DSN = DSNDXW.DXW2.LOGCOPY1.DS04.DATA
BMC40566I  USING LOG DSNDXW.DXW1.LOGCOPY1.DS02.DATA FOR PROCESSING
BMC40555I       FROMRBA  = X'000000000221FB0DF000'  TORBA  = X'000000000222060A6FFF'
BMC40566I  USING LOG DSNDXW.DXW3.LOGCOPY1.DS02.DATA FOR PROCESSING
BMC40555I       FROMRBA  = X'00000000018750C8C000'  TORBA  = X'0000000001875BC53FFF'
BMC40566I  USING LOG DSNDXW.DXW4.LOGCOPY1.DS01.DATA FOR PROCESSING
BMC40555I       FROMRBA  = X'00000000015B8AF7D000'  TORBA  = X'00000000015B95F44FFF'
BMC40566I  USING LOG DSNDXW.DXW2.LOGCOPY1.DS04.DATA FOR PROCESSING
BMC40555I       FROMRBA  = X'0000000001C85AD13000'  TORBA  = X'0000000001C91C2D2FFF'
BMC40189I  THE 0 TAPE AND 4 DISK LOG FILES WILL BE READ IN 2 LOG READING STEPS BECAUSE
MAXDRIVES = 0 AND MAXLOGS = 3
BMC40335I  PLAN EXECUTION STARTING 08/17/2012 13:39:22
BMC40025I  OUTPUT FROM COMMAND '-STA DB(BMCDBSMP) SPACE(BMCTS011) ACCESS(RW)' FOLLOWS:
BMC40020I  DSN9022I  *DXW4 DSNTTDIS 'START DATABASE' NORMAL COMPLETION
BMC40025I  OUTPUT FROM COMMAND '-STA DB(BMCDBSMP) SPACE(BMCIX011) ACCESS(RW)' FOLLOWS:
BMC40020I  DSN9022I  *DXW4 DSNTTDIS 'START DATABASE' NORMAL COMPLETION

Figure 85: Example 2B--AFRPRINT for a simple RECOVER INDEX

BMC40018I  RECOVER PLUS FOR DB2 V11.1.00
BMC40473I  COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC40189I  LOG INPUT PHASE MESSAGES ARE IN FILE AFRPR001
BMC40189I  MERGE PHASE MESSAGES FOR THE FOLLOWING OBJECT ARE IN FILE AFRPR001
BMC40303I  INDEX = RMD.BMCIX011
BMC40189I  MERGE PHASE MESSAGES FOR THE FOLLOWING OBJECT ARE IN FILE AFRPR002
BMC40301I  TABLESPACE = BMCDBSMP.BMCTS011

Figure 86: Example 2B--AFRPR001 for a simple RECOVER INDEX

BMC40016I  LOG INPUT PHASE STARTING 08/17/2012 13:39:22
BMC40954I  AVAILABLE MEMORY BELOW THE LINE = 10044K, TOTAL AVAILABLE MEMORY (AT LEAST) = 
1370276K
BMC40441I  BMCSORT PRODUCT LOCATED AND WILL BE USED
BMC40020I  ORG-BEL:0, 0, 0; EXT:0, 0, 0; TOT:857488, 5112516, 5112516; ITER: 1; RC:4
BMC40020I  SYSEVENT STGTEST STORAGE AVAILABLE = 5112516
BMC40954I  AVAILABLE MEMORY BELOW THE LINE = 10044K, TOTAL AVAILABLE MEMORY (AT LEAST) = 
1370276K
BMC40441I  BMCSORT PRODUCT LOCATED AND WILL BE USED
BMC40020I  ORG-BEL:0, 0, 0; EXT:0, 0, 0; TOT:857488, 5112516, 5112516; ITER: 1; RC:4
```
**Example 2: Recovering to the current state**

Chapter 5 Examples of NGT Recover jobs 389
Example 2: Recovering to the current state

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC401775I</td>
<td>390 WAITS FOR THREAD WAIT TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC401241I</td>
<td>0 SORT WAITS SORT WAIT TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40172I</td>
<td>0 LOG RECORDS AND 0 BYTES OF LOG DATA PROCESSED</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = DSNDXW.DXW2.LOGCOPY1.DS04.DATA</td>
</tr>
<tr>
<td>BMC40174I</td>
<td>TIME SPENT IN DATASET ALLOCATION = 00:00:00</td>
</tr>
<tr>
<td>BMC40174I</td>
<td>6255 I/O WAITS I/O WAIT TIME = 00:00:05</td>
</tr>
<tr>
<td>BMC40172I</td>
<td>0 LOG RECORDS AND 0 BYTES OF LOG DATA PROCESSED</td>
</tr>
<tr>
<td>BMC40124I</td>
<td>0 SORT WAITS SORT WAIT TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40372I</td>
<td>TIME SPENT IN DATASET ALLOCATION = 00:00:00</td>
</tr>
<tr>
<td>BMC40172I</td>
<td>0 LOG RECORDS AND 0 BYTES OF LOG DATA PROCESSED</td>
</tr>
<tr>
<td>BMC40012I</td>
<td>LOG INPUT PHASE COMPLETE. ELAPSED TIME = 00:00:13, TIME SINCE UTILITY START = 00:00:18</td>
</tr>
<tr>
<td>BMC40868I</td>
<td>LOG INPUT PHASE COMPLETE. ACCUMULATED TCB TIME = 00:00:01</td>
</tr>
<tr>
<td>BMC40484I</td>
<td>MESSAGES FROM PHASE SUBTASK:</td>
</tr>
<tr>
<td>BMC40350I</td>
<td>DATA SET DSNDXW.DXW2.LOGCOPY1.DS02.DATA WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00010 ELAPSED = 00:00:00</td>
</tr>
<tr>
<td>BMC40350I</td>
<td>DATA SET DSNDXW.DXW4.LOGCOPY1.DS01.DATA WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00011 ELAPSED = 00:00:00</td>
</tr>
<tr>
<td>BMC40350I</td>
<td>DATA SET DSNDXW.DXW1.LOGCOPY1.DS02.DATA WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00012 ELAPSED = 00:00:00</td>
</tr>
<tr>
<td>BMC40350I</td>
<td>DATA SET DSNDXW.DXW2.LOGCOPY1.DS04.DATA WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00017 ELAPSED = 00:00:00</td>
</tr>
<tr>
<td>BMC40485I</td>
<td>END OF PHASE SUBTASK MESSAGES</td>
</tr>
<tr>
<td>BMC40016I</td>
<td>MERGE PHASE STARTING 08/17/2012 13:39:35</td>
</tr>
<tr>
<td>BMC40345I</td>
<td>SPACE = BMCDBSMP.BMCIX011</td>
</tr>
<tr>
<td>BMC40356I</td>
<td>AN IDCAMS 'DELETE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DSNDXW.DSNDBC.BMCDBSMP.BMCIX011.I0001.A001</td>
</tr>
<tr>
<td>BMC40356I</td>
<td>AN IDCAMS 'DEFINE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DSNDXW.DSNDBC.BMCDBSMP.BMCIX011.I0001.A001</td>
</tr>
<tr>
<td>BMC40350I</td>
<td>DATA SET DSNDXW.DSNDBD.BMCDBSMP.BMCIX011.I0001.A001 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00016 ELAPSED = 00:00:00</td>
</tr>
<tr>
<td>BMC40485I</td>
<td>END OF PHASE SUBTASK MESSAGES</td>
</tr>
<tr>
<td>BMC40350I</td>
<td>DATA SET RDAPXR.BMCDBSMP.BMCIX011.F.T133832 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00018 ELAPSED = 00:00:00</td>
</tr>
<tr>
<td>BMC40394I</td>
<td>THIS PHASE WILL RUN IN EXECUTION SUBTASK 1</td>
</tr>
<tr>
<td>BMC40593I</td>
<td>START MERGE COMPLETE, ELAPSED TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40102I</td>
<td>MERGE INDEXSPACE STEP STARTING</td>
</tr>
<tr>
<td>BMC40795I</td>
<td>FULL COPY DATASET IS RDAPXR.BMCDBSMP.BMCIX011.F.T133832</td>
</tr>
<tr>
<td>BMC40799I</td>
<td>MERGE STARTING FOR DATA SET DSNDXW.DSNDBD.BMCDBSMP.BMCIX011.I0001.A001</td>
</tr>
<tr>
<td>BMC40593I</td>
<td>INITIALIZATION COMPLETE, ELAPSED TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40088I</td>
<td>WAIT TIME FOR SORT OUTPUT PHASE = 00:00:00</td>
</tr>
<tr>
<td>BMC40593I</td>
<td>SPIN COMPLETE, ELAPSED TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40902I</td>
<td>TOTAL WAITS = 3, TOTAL WAIT TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40981I</td>
<td>FULL IC RDAPXR.BMCDBSMP.BMCIX011.F.T133832 READ WAITS = 1, WAIT TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40979I</td>
<td>LOG INPUT WAITS = 1, WAIT TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40905I</td>
<td>SPACE WRITE WAITS = 1, WAIT TIME = 00:00:00</td>
</tr>
<tr>
<td>BMC40985I</td>
<td>NUMBER OF PAGES READ FROM FULL COPY FOR THIS OBJECT = 20, USED = 20</td>
</tr>
<tr>
<td>BMC40791I</td>
<td>NUMBER OF PAGES READ FROM FULL COPY TOTAL = 20, USED = 20</td>
</tr>
<tr>
<td>BMC40981I</td>
<td>LOG RECORDS FROM SORT = 0 (0 BYTES), SELECTED = 0</td>
</tr>
<tr>
<td>BMC40981I</td>
<td>LOG RECORDS APPLIED = 0</td>
</tr>
<tr>
<td>BMC40981I</td>
<td>LOG RECORDS SELECTED = 0</td>
</tr>
<tr>
<td>BMC40981I</td>
<td>NUMBER OF PAGES OUTPUT IS = 20</td>
</tr>
</tbody>
</table>
Example 2: Recovering to the current state

Chapter 5 Examples of NGT Recover jobs 391
Figure 87: Example 2B--AFRPR002 for a simple RECOVER INDEX

BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40016I MERGE PHASE STARTING 08/17/2012 13:39:35
BMC40301I TABLESPACE = BMCDBSMP.BMCTS011
BMC40484I MESSAGES FROM PHASE SUBTASK:
BMC40060I PENDING FLAG RECP SET FOR
BMC40345I SPACE = BMCDBSMP.BMCTS011
BMC40356I AN IDCAMS 'DELETE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001
BMC40356I AN IDCAMS 'DEFINE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001
BMC40356I DATA SET DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00014 ELAPSED = 00:00:00
BMC40417I 2 VSAM DATA SETS NOW ALLOCATED
BMC40485I END OF PHASE SUBTASK MESSAGES
BMC40350I DATA SET RDAPXR.BMCDBSMP.BMCTS011.F.T133831 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00019 ELAPSED = 00:00:00
BMC96304I THIS PHASE WILL RUN IN EXECUTION SUBTASK 1
BMC40593I START MERGE COMPLETE, ELAPSED TIME = 00:00:00
BMC40102I MERGE TABLESPACE STEP STARTING
BMC40795I FULL COPY DATASET IS RDAPXR.BMCDBSMP.BMCTS011.F.T133831
BMC40799I MERGE STARTING FOR DATA SET DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001
BMC40593I INITIALIZATION COMPLETE, ELAPSED TIME = 00:00:00
BMC40088I WAIT TIME FOR SORT OUTPUT PHASE = 00:00:00
BMC40593I SPIN COMPLETE, ELAPSED TIME = 00:00:00
BMC40401I EXTENT ALLOCATED FOR TABLE SPACE BMCDBSMP.BMCTS011
BMC40302I DSN = DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001
BMC40401I EXTENT ALLOCATED FOR TABLE SPACE BMCDBSMP.BMCTS011
BMC40302I DSN = DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001
BMC40401I EXTENT ALLOCATED FOR TABLE SPACE BMCDBSMP.BMCTS011
BMC40302I DSN = DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001
BMC40902I TOTAL WAITS = 9, TOTAL WAIT TIME = 00:00:00
BMC40981I FULL IC RDAPXR.BMCDBSMP.BMCTS011.F.T133831 READ WAITS = 1, WAIT TIME = 00:00:00
BMC40979I LOG INPUT WAITS = 1, WAIT TIME = 00:00:00
BMC40905I SPACE WRITE WAITS = 7, WAIT TIME = 00:00:00
BMC40985I NUMBER OF PAGES READ FROM FULL COPY FOR THIS OBJECT = 2503, USED = 2503
BMC40790I NUMBER OF PAGES READ FROM FULL COPY TOTAL = 2503, USED = 2503
BMC40792I LOG RECORDS THIS PAGESET: FROM SORT = 12498 ( 819K BYTES), SELECTED = 12498
BMC96081I APPLIED = 12498
BMC40989I LOG RECORDS TOTAL: FROM SORT = 12498 ( 819K BYTES), SELECTED = 12498
BMC96081I APPLIED = 12498
BMC40798I NUMBER OF PAGES OUTPUT IS = 2503
BMC40900I MERGE TABLESPACE STEP COMPLETE
BMC40593I MERGE PROCESS COMPLETE, ELAPSED TIME = 00:00:00
BMC40352I DDNAME SYS00014 (DSN DSNDXW.DSNDBD.BMCDBSMP.BMCTS011.I0001.A001) WAS SUCCESSFULLY UNALLOCATED
BMC40352I DDNAME SYS00019 (DSN RDAPXR.BMCDBSMP.BMCTS011.F.T133831) WAS SUCCESSFULLY UNALLOCATED
BMC40091I 08/17/2012 13:39:35 SUBPHASE CHECKPOINT TAKEN FOR MERGE PHASE
BMC40012I MERGE PHASE COMPLETE, ELAPSED TIME = 00:00:00, TIME SINCE UTILITY START = 00:00:19
BMC40868I MERGE PHASE COMPLETE, ACCUMULATED TCB TIME = 00:00:01
Figure 88: AFROSUM for a simple RECOVER INDEX

BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND
B.161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40709I OBJECT SUMMARY:
BMC40301I TABLESPACE = BMCDBSMP.BMCTS011
BMC40711I DBID = X'0267' PSID = X'000E'
BMC40712I ORIGINAL STATUS = RW
BMC40716I STOGROUP = SYSDEFLT PRIQTY = 1200 SECQTY = 400
BMC40710I PAGE SIZE = 4K SEGSIZE = 4
BMC40723I ENCODING SCHEME = EBCDIC

BMC40820I TABLESPACE RECOVERY
BMC40827I LOG RESOURCE RANGE:
BMC40585I FROM LOGPOINT = X'00CA08AC036260000000' TO LOGPOINT =
X'00CA08AC3005F5000000'
BMC40727I DISCRETE LOG RANGE(S):
BMC40559I FROMLRSN = X'00CA08AC0434E1000000' TOLRSN =
X'00CA08AC04924C000000' MEMBER = DXW3 ID = 2
BMC40555I FROMRBA = X'00000000018753B91F0' TORBA = X'00000000018753C9AF9'

BMC40840I INPUT COPY / SYSCOPY DATA:
BMC40640I 2012-08-17 13.38.31.4241 RBA/LRSN X'00CA08AC036260000000' FULL IMAGE
COPY
BMC40302I DSN = RDAPXR.BMCDBSMP.BMCTS011.F.T133831
BMC40641I SHRLEVEL = REFERENCE SITETYPE = LP DSNUM = 0
BMC40401I 2012-08-17 13.38.31.5971 RBA/LRSN X'00CA08AC03BB46000000' QUIESCE
UTILITY POINT - WRITE(YES)
BMC40641I 2012-08-17 13.38.32.4810 RBA/LRSN X'00CA08AC04930E000000' QUIESCE
UTILITY POINT - WRITE(YES)

BMC40709I OBJECT SUMMARY:
BMC40346I INDEXSPACE = BMCDBSMP.BMCIX011
BMC40711I DBID = X'0267' PSID = X'0011'
BMC40712I ORIGINAL STATUS = RW
BMC40303I INDEX = RMD.BMCIX011
BMC96160I PAGE SIZE = 4K
BMC40716I STOGROUP = SYSDEFLT PRIQTY = 60 SECQTY = 40
BMC40830I ON TABLE RMD.BMCTB011 (OBID X'000F') IN TABLESPACE BMCDBSMP.BMCTS011
BMC40836I TABLE CARDINALITY = -1
BMC40832I NON-CLUSTERING INDEX
BMC40833I UNIQUE INDEX
BMC40844I TYPE 2
BMC40843I INDEX RECOVERY
BMC40827I LOG RESOURCE RANGE:
BMC40585I FROM LOGPOINT = X'00CA08AC03CEDC000000' TO LOGPOINT =
X'00CA08AC3005F5000000'
BMC40727I DISCRETE LOG RANGE(S):
BMC40559I FROMLRSN = X'00CA08AC03CEDC000000' TOLRSN =
X'00CA08AC3005F5000000' MEMBER = DXW1 ID = 1
BMC40555I FROMRBA = X'00000000000000000000' TORBA = X'FFFFFFFFFFFFFFFFFFFF'
BMC40559I FROMLRSN = X'00CA08AC03CEDC000000' TOLRSN =
X'00CA08AC3005F5000000' MEMBER = DXW3 ID = 2
BMC40555I FROMRBA = X'00000000000000000000' TORBA = X'FFFFFFFFFFFFFFFFFFFF'
BMC40559I FROMLRSN = X'00CA08AC03CEDC000000' TOLRSN =
X'00CA08AC3005F5000000' MEMBER = DXW4 ID = 3
Example 2: Recovering to the current state

Figure 89: Example 2B—AFRPLAN for a simple RECOVER INDEX

BMC40018I RECOVER PLUS FOR DB2 V11.1.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2013
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40700I EXECUTION PLAN SUMMARY:

BMC40701I PHASE: LOG INPUT
BMC40721I PHASE INITIALIZATION:
BMC40738I INITIALIZE LOG RECORD SORT 1
BMC40814I ESTIMATED RECORD SIZE = 1000
BMC40813I ESTIMATED NUMBER OF RECORDS = 322
BMC40738I ALLOCATE 2 POINTER BUFFERS (53092 BYTES EACH)
BMC40738I INITIALIZE LOG RECORD SORT 2
BMC40814I ESTIMATED RECORD SIZE = 1000
BMC40813I ESTIMATED NUMBER OF RECORDS = 1236000
BMC40738I ALLOCATE 2 POINTER BUFFERS (53092 BYTES EACH)

BMC40702I STEP: READ LOG PAGES
BMC40736I STEP INITIALIZATION:
BMC40738I ALLOCATE INPUT LOG FILE
BMC40302I DSN = DSNDXW.DXW4.LOGCOPY1.DS01.DATA
BMC40738I ALLOCATE INPUT LOG FILE
BMC40302I DSN = DSNDXW.DXW3.LOGCOPY1.DS02.DATA
BMC40738I ALLOCATE INPUT LOG FILE
BMC40302I DSN = DSNDXW.DXW1.LOGCOPY1.DS02.DATA
BMC40741I STEP PROCESSING:
BMC40738I FILE OPTIMIZATION
BMC40738I OPEN LOG DATASET
BMC40302I DSN = DSNDXW.DXW4.LOGCOPY1.DS01.DATA
BMC40738I ALLOCATE LOG DATASET BUFFER(S)
BMC40738I PREPARE LOG FOR PROCESSING
BMC40302I DSN = DSNDXW.DXW4.LOGCOPY1.DS01.DATA
BMC407041 45000 PAGES FROM PAGE NUMBER 0
BMC40738I FILE OPTIMIZATION
BMC40738I OPEN LOG DATASET
BMC40302I DSN = DSNDXW.DXW3.LOGCOPY1.DS02.DATA
BMC40738I ALLOCATE LOG DATASET BUFFER(S)
BMC40738I PREPARE LOG FOR PROCESSING
BMC40302I DSN = DSNDXW.DXW3.LOGCOPY1.DS02.DATA
BMC407041 45000 PAGES FROM PAGE NUMBER 0
BMC40738I FILE OPTIMIZATION
BMC40738I OPEN LOG DATASET
BMC40302I DSN = DSNDXW.DXW1.LOGCOPY1.DS02.DATA
BMC40738I ALLOCATE LOG DATASET BUFFER(S)
BMC40738I PREPARE LOG FOR PROCESSING
BMC40302I DSN = DSNDXW.DXW1.LOGCOPY1.DS02.DATA
Example 2: Recovering to the current state

Chapter 5  Examples of NGT Recover jobs  395
Example 2: Recovering to the current state

BMC40301I   TABLESPACE = BMCDBSMP.BMCTS011
BMC40735I   PHASE TERMINATION:
BMC40738I    DISCONNECT FROM SPACE
BMC40738I    DEALLOCATE SPACE
BMC40345I    SPACE = BMCDBSMP.BMCTS011
BMC40738I    FREE PAGE CHECK BUFFER
BMC40738I    FREE SPACEMAP BUFFER
BMC40738I    FREE PASTE BUFFER
BMC40738I    FREE 2 OUTPUT SPACE BUFFER(S)
BMC40738I    FREE PAGE CHECK BUFFER
BMC40738I    FREE SPACEMAP BUFFER
BMC40738I    CLOSE IMAGE COPY FILE
BMC40302I    DSN = RDAPXR.BMCDBSMP.BMCTS011.F.T133831
BMC40738I    FREE 2 INPUT FILE BUFFER(S)
BMC40738I    DEALLOCATE INPUT IMAGE COPY FILE
BMC40302I    DSN = RDAPXR.BMCDBSMP.BMCTS011.F.T133831
BMC40738I    RESET RECOVERY PENDING (RECP) FLAG
BMC40738I    END LOG RECORD SORT
BMC40738I    FREE 3 LOG RECORD BUFFER(S)
BMC40738I    REGISTER RECOVER TO CURRENT IN SYSCOPY
BMC40738I    RECORD RESTART INFORMATION

BMC40701I   PHASE: MERGE
BMC40721I    PHASE INITIALIZATION:
BMC40303I    INDEX = RMD.BMCIX011
BMC40738I    ALLOCATE 3 LOG RECORD BUFFER(S) (81920 BYTES EACH)
BMC40738I    ALLOCATE INPUT IMAGE COPY FILE
BMC40302I    DSN = RDAPXR.BMCDBSMP.BMCIX011.F.T133832
BMC40738I    FILE OPTIMIZATION
BMC40738I    ALLOCATE 2 INPUT FILE BUFFER(S)
BMC40738I    OPEN IMAGE COPY FILE
BMC40302I    DSN = RDAPXR.BMCDBSMP.BMCIX011.F.T133832
BMC40738I    ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I    ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I    ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40738I    SET REBUILD PENDING (RBDP) FLAG
BMC40738I    CREATE SPACE
BMC40345I    SPACE = BMCDBSMP.BMCIX011
BMC40738I    ALLOCATE SPACE (READ/WRITE)
BMC40345I    SPACE = BMCDBSMP.BMCIX011
BMC40738I    CONNECT TO SPACE
BMC40738I    ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I    FILE OPTIMIZATION
BMC40738I    ALLOCATE 2 SPACE OUTPUT BUFFER(S)
BMC40738I    ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I    ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40702I     STEP: MERGE SPACE
BMC40738I     MERGE SPACE
BMC40303I     INDEX = RMD.BMCIX011
BMC40735I     PHASE TERMINATION:
BMC40738I     DISCONNECT FROM SPACE
BMC40738I     DEALLOCATE SPACE
BMC40345I     SPACE = BMCDBSMP.BMCIX011
BMC40738I     FREE PAGE CHECK BUFFER
BMC40738I     FREE SPACEMAP BUFFER
BMC40738I     FREE PASTE BUFFER
BMC40738I     FREE 2 OUTPUT SPACE BUFFER(S)
BMC40738I     FREE PAGE CHECK BUFFER
BMC40738I     FREE PASTE BUFFER
BMC40738I     FREE SPACEMAP BUFFER
BMC40738I     CLOSE IMAGE COPY FILE
BMC40302I     DSN = RDAPXR.BMCDBSMP.BMCIX011.F.T133832
BMC40738I     FREE 2 INPUT FILE BUFFER(S)
BMC40738I     DEALLOCATE INPUT IMAGE COPY FILE
BMC40302I     DSN = RDAPXR.BMCDBSMP.BMCIX011.F.T133832
BMC40738I     RESET REBUILD PENDING (RBDP) FLAG

396   BMC Next Generation Technology Recover for DB2 for z/OS Reference Manual
Example 3: Creating and dynamically allocating output copies

In the following example, NGT Recover recovers one partition of the table space and creates one image copy. NGT Recover uses dynamic allocation for the image copy.

The OUTPUT command is used to dynamically allocate the copy that is made.

Figure 90: Example 3—JCL for recovering one partition and creating an image copy

Example 4: Recovering to a specified copy

With the TOCOPY option, you can specify the name of the last image copy to be used by NGT Recover when restoring the table space.

If the copy that you specify is an incremental image copy, the prior full copy and any intervening incremental image copies are merged.
Example 4A: Using TOCOPY with a named copy

The example shown in the following figure illustrates a table space recovery using only a full image copy and an index rebuild that includes all of the indexes on the table space.

**Figure 91: Example 4A--JCL using TOCOPY with a named copy**

```plaintext
//AFREX04A JOB (PAFR), 'EXAMPLE 4A',
//   CLASS=Q, NOTIFY=&SYSUID,
//   MSGCLASS=X
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=0M,
//              PARM='DGE, EXAMPLE04A, NEW/RESTART, MSGLEVEL(2)'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//       DD DISP=SHR, DSN=DB2.DSNEXIT
//       DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN  DD *
RECOVER TABLESPACE BMCDBSMP.BMCTS014
   TOCOPY RDASTC.BMCDBSMP.BMCTS014.IC(0)
   REBUILD INDEX(ALL) TABLESPACE BMCDBSMP.BMCTS014
   NOWORKDDN
/**
//*
```

Example 4B: Using TOCOPY LASTCOPY

The example shown in the following figure uses the keyword LASTCOPY for the value of the TOCOPY data set name instead of providing the name of the copy.

Using TOCOPY LASTCOPY can be useful in a number of routine situations. In the example, a routine backup of three table spaces is performed (using NGT Copy) followed by a batch application that updates those table spaces with transactions that have accumulated since the prior backup. This process is followed by a conditional recovery.

If the batch application:

- Ends abnormally, the spaces are restored to the state existing at the time the current copy was made using TOCOPY LASTCOPY instead of naming the copy
- Runs successfully, no recovery is performed.

Using TOCOPY LASTCOPY in this way avoids having to modify the JCL for routine operations.

**Figure 92: Example 4B--JCL using TOCOPY LASTCOPY**

```plaintext
//AFREX04B JOB (PAFR), 'EXAMPLE 4B',
//   CLASS=Q, NOTIFY=&SYSUID,
//   MSGCLASS=X
//*
//* -------------------------------------------------------------------
```

Example 4: Recovering to a specified copy
THIS STEP TAKES FULL IMAGE COPIES OF THREE TABLE SPACES

//BMCCOPY EXEC PGM=ACPMAIN,REGION=OM,
// PARM='DGE,COPY4B,NEW/RESET,MSGLEVEL(1)'
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//CP00001 DD DSN=RDASTC.BMCDBSMP.BMCTS001.IC(+1),DISP=(,CATLG),
// DCB=(SYS1.MODEL),UNIT=SYSDA,SPACE=(CYL,(25,5))
//CP00002 DD DSN=RDASTC.BMCDBSMP.BMCTS002.IC(+1),DISP=(,CATLG),
// DCB=(SYS1.MODEL),UNIT=SYSDA,SPACE=(CYL,(25,5))
//CP00003 DD DSN=RDASTC.BMCDBSMP.BMCTS003.IC(+1),DISP=(,CATLG),
// DCB=(SYS1.MODEL),UNIT=SYSDA,SPACE=(CYL,(25,5))
//SYSIN DD *
COPY TABLESPACE BMCDBSMP.BMCTS001
  FULL YES COPYDDN (CP00001) RESETMOD YES
  SHRLEVEL REFERENCE
COPY TABLESPACE BMCDBSMP.BMCTS002
  FULL YES COPYDDN (CP00002) RESETMOD YES
  SHRLEVEL REFERENCE
COPY TABLESPACE BMCDBSMP.BMCTS003
  FULL YES COPYDDN (CP00003) RESETMOD YES
  SHRLEVEL REFERENCE
/*
//SYSPRINT DD SYSOUT=* 
/*

THIS STEP UPDATES THE TABLE SPACES COPIES IN THE PREVIOUS STEP.

//UPDATER EXEC PGM=
//...................................................................
//... (BATCH APPLICATION THAT UPDATES THE COPIED TABLE SPACES) ......
//... (SIMULATED WITH FOLLOWING SQL) ......................................
//...................................................................
// UPDATE BMCTB001, BMCTB002, BMCTB003
//==================================================================
//UPDATE01 EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=SYS2.DB2.DSNLOAD,DISP=SHR
//SYSTSRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSIN DD *
// DSN SYSTEM(DGE)
// RUN PROGRAM(BMCTEP2)
//END
/*
//SYSSIN DD *
UPDATE RDASTC.BMCTB001 SET COL2 = 'UPDATE1';
UPDATE RDASTC.BMCTB001 SET COL2 = 'UPDATE2';
//...................................................................
//UPDATE02 EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=SYS2.DB2.DSNLOAD,DISP=SHR
//SYSTSRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSIN DD *
// DSN SYSTEM(DGE)
// RUN PROGRAM(BMCTEP2)
//END
/*
//SYSSIN DD *
UPDATE RDASTC.BMCTB002 SET COL2 = 'UPDATE1';
UPDATE RDASTC.BMCTB002 SET COL2 = 'UPDATE2';
//...................................................................
//UPDATE03 EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=SYS2.DB2.DSNLOAD,DISP=SHR
Example 5: Recovering to a specific log point

This example illustrates two ways of recovering all parts of a table space and its indexes to a prior point in time by specifying TOLOGPOINT with the keyword LASTQUIESCE. This method allows a recovery to the last quiesce point registered for the table space in SYSIBM.SYSCOPY.

The log point specified by TOLOGPOINT LASTQUIESCE defines the point at which recovery stops. Only log records with starting log points equal to or lower than LASTQUIESCE and only image copies with log points less than LASTQUIESCE are used in the recovery.
Example 5A Using the TOLOGPOINT option with RECOVER

Example 5A uses image copies and archive and active logs to recover all partitions of the table space.

The indexes are also recovered from image copies and logs. A SYSPICK DD statement is included to obtain a list of all of the input tape and cartridge volumes that are allocated during recovery.

Figure 93: Example 5A--JCL using TOLOGPOINT with RECOVER

```
//AFREX05A JOB (PAFR), 'EXAMPLE 5A',
//         CLASS=Q, NOTIFY=&SYSUID,
//         MSGCLASS=X
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=0M,
//         PARM='DGE,EXAMPLE5A,NEW/RESTART,MSGLEVEL(2)'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//         DD DISP=SHR, DSN=DB2.DSNEXIT
//         DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
// OPTIONS INDEXLOG YES
// RECOVER TABLESPACE BMCDBSMP.BMCTS012
//TOLOGPOINT LASTQUIESCE
// RECOVER INDEX(ALL) TABLESPACE BMCDBSMP.BMCTS012
//*
//SYSPICK DD SYSOUT=*  
//*
```

Example 5B: Using TOLOGPOINT and BACKOUT

Example 5B uses the BACKOUT feature to recover the table space and its indexes.

Figure 94: Example 5B--JCL using TOLOGPOINT and BACKOUT

```
//AFREX05B JOB (PAFR), 'EXAMPLE 5B',
//         CLASS=Q, NOTIFY=&SYSUID,
//         MSGCLASS=X
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=0M,
//         PARM='DGE,EXAMPLE5B,NEW/RESTART,MSGLEVEL(2)'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//         DD DISP=SHR, DSN=DB2.DSNEXIT
//         DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
// OPTIONS INDEXLOG YES
// RECOVER TABLESPACE BMCDBSMP.BMCTS013
// TOLOGPOINT LASTQUIESCE
// BACKOUT
// RECOVER INDEX(ALL) TABLESPACE BMCDBSMP.BMCTS013
//*
```
Example 6: Extracting nonpartitioned index keys

Example 6 illustrates the recovery of parts 1 and 2 of a 6 partition table space, the recovery of parts 1 and 2 of its partitioned index, and the unloading of keys from those partitions for the nonpartitioned index.

The unloaded keys are sorted before being written to the work file. Notice that there is only one work file in this job and that the DDNAME for the work file defaults to SKEY. The keys for partitions 3 through 6 are unloaded and sorted in two other similar jobs, which can run concurrently with this one. The output from these jobs is used in “Example 7: Building a nonpartitioned index” on page 402 to build an index.

Figure 95: Example 6--JCL recovering a partitioned table space and partitioned index and unloading keys for a nonpartitioned index

Example 7: Building a nonpartitioned index

Example 7 uses the keys that were unloaded and sorted in Example 6 to build an index.

See “Example 6: Extracting nonpartitioned index keys” on page 402.
Example 8: Recovering using tape-stacked data sets

Example 8 consists of two jobs. The first job uses the NGT Copy utility to stack three table spaces to different data sets on the same tape volume. The second job uses NGT Recover to recover those table spaces.

*Tape stacking* refers to copying several small table spaces or table space partitions to different data sets on one or more tape volumes. The advantage of tape stacking is that it reduces the number of tapes needed for your image copies.

NGT Recover dynamically allocates the data set and the tape volumes and avoids excessive rewinding and positioning. You do not need to specify the data sets in the JCL.

**Example 8A: Making stacked image copies**

The JCL in the following figure makes one full image copy of each of three table spaces, registers them, and writes them to the same cartridge tape.

Figure 97: Example 8A--JCL for making stacked image copies

```
//AFREX8A JOB (PAFR),'EXAMPLE 8A', //
//       CLASS=Q,NOTIFY=&SYSUID, //
//       MSGCLASS=X //
/*
//ACPCOPY EXEC PGM=ACPMAIN,REGION=0M, //
//       PARM='DGE,EXAMPLE8A,NEW/RESET' //
//STEPLIB DD DISP=SHR,DSN=product.libraries //
//       DD DISP=SHR,DSN=DB2.DSNEXIT //
//       DD DISP=SHR,DSN=DB2.DSNLOAD //
//SKEY1 DD DISP=SHR,DSN=RDASTC.BMCDBJ.PARTS12.SUNKEY1 //
//SKEY2 DD DISP=SHR,DSN=RDASTC.BMCDBJ.PARTS34.SUNKEY2 //
//SKEY3 DD DISP=SHR,DSN=RDASTC.BMCDBJ.PARTS56.SUNKEY3 //
//SYSIN DD * //
/*
//RECOVER BUILDINDEX (RDASTC.BMCIXJ2) /*
//
```
Example 8B: Recovering with stacked image copies

The JCL in the following figure performs a recovery using the three image copies.

**Note**

In this example, you do not need to order the RECOVER TABLESPACE statements. NGT Recover orders the activities to use the data sets sequentially.

**Figure 98: Example 8B—JCL for recovering with stacked cataloged image copies as input**

Example 9: Recovering to a non-DB2 data set

Recovering to data sets not used by DB2 is useful for testing or for setting up recovered spaces to switch to while READ ONLY access continues to the real data sets.
Although INDEP INTABLESPACE is not explicitly specified with REBUILD INDEX in the SYSIN statement in the recovery JCL in the following figure, NGT Recover assumes it and defaults to the RECOVER TABLESPACE non-DB2 data set for index data input.

**Figure 99: Example 9—JCL for simple table space and index recovery to non-DB2 data sets**

```
//AFREX10 JOB (PAFR), 'EXAMPLE 9',
//   CLASS=Q, NOTIFY=&SYSUID,
//   MSGCLASS=X
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=0M,
//   PARM='DEBA,EXAMPLE9,NEW/RESTART,MSGLEVEL(2)'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//   DD DISP=SHR, DSN=DB2.DSNEXIT
//   DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
RECOVER TABLESPACE BMCDBSMP.BMCCTS010 INDEP OUTSPACE
REBUILD INDEX (RMD.BMCIX010) INDEP OUTSPACE NOWORKDDN
/*
```

**Directing recovery output**

When you use the NGT Recover redirection feature to direct the output of a table space or index recovery to a data set other than the one normally used by DB2 or to extract index information from such a data set, you must construct your RECOVER statements according to the following rules:

- When you specify redirection of the recovery output in a RECOVER TABLESPACE statement, you append the keyword INDEPENDENT (which you can abbreviate to INDEP) followed by OUTSPACE and (optionally) the model data set name (`dataSetName`) to the statement as follows:

  ```
  RECOVER TABLESPACE `tableSpaceSpecification'
  INDEP OUTSPACE `optional dataSetName model`
  other recovery options
  ```

  **Note**
  NGT Recover requires consistent use of the INDEP OUTSPACE option across all commands; if you use INDEP OUTSPACE with one command, you must use it with all other commands that support it. If you use INDEP OUTSPACE but subsequently omit it from another command that supports it, NGT Recover issues an error message.

- When you specify redirection of the recovery output in a REBUILD INDEX or RECOVER BUILDINDEX statement, you insert INDEP OUTSPACE before the table space specification, as follows:

  ```
  REBUILD INDEX ... INDEP OUTSPACE `optional dataSetName model`
  optional `tableSpaceSpecification`
  other recovery options
  RECOVER BUILDINDEX ... INDEP OUTSPACE
  ```
When you rebuild indexes from data that resides in an independent data set, you must also redirect the input for the index rebuild as follows:

```
REBUILD INDEX ... INDEP OUTSPACE optional dataSetName model
optional tableSpaceSpecification
INDEP INTABLESPACE optional dataSetName model
other recovery options
```

In addition, if the SYSIN statement in which the REBUILD INDEX appears also includes a RECOVER TABLESPACE statement that specifies redirection, any data set name model specified with INDEP INTABLESPACE must agree with the RECOVER TABLESPACE data set name model. If you do not specify INDEP INTABLESPACE explicitly, NGT Recover assumes it and defaults to the RECOVER TABLESPACE data set name model for index data input.

NGT Recover does not allow:

— The recovery of indexes into DB2 data sets when the corresponding table space recovery is to a non-DB2 (independent) data set

— The recovery of indexes into DB2 data sets when the source data for the table space is a non-DB2 data set

That is, if you specify INDEP INTABLESPACE in a REBUILD INDEX statement, you must also specify INDEP OUTSPACE. Similarly, if an index rebuild implies an INDEP INTABLESPACE (because it is in the same SYSIN statement as a RECOVER TABLESPACE with an INDEP OUTSPACE clause for the related table space), the REBUILD INDEX must also have an INDEP OUTSPACE clause.

If you are extracting index keys (using a RECOVER UNLOADKEYS statement) from data that has already been redirected to an independent data set, you insert INDEP INTABLESPACE after the table space specification as follows:

```
RECOVER UNLOADKEYS ... optional table space
 specification
INDEP INTABLESPACE optional dataSetName model
other recovery options
```

The following figures show the following output files produced by this JCL:

— AFRSUMRY
— AFRSTMT
— AFRPRINT
— AFRPR001
— AFRPR002
— AFROSUM
— AFRPLAN

**Figure 100: Example 9—AFRSUMRY for recovering to a non-DB2 data set**
Figure 101: Example 9—AFRPRINT for recovering to a non-DB2 data set

Example 9: Recovering to a non-DB2 data set

Figure 102: Example 9—AFRPR001 for recovering to a non-DB2 data set
Example 9: Recovering to a non-DB2 data set
Prefetch count = 0, Hits caused by Prefetch = 0, Hits on Prefetch = 0
Total wait time = 00:00:00, Random write time = 00:00:00, Random read time = 00:00:00
Cylinder wait time = 00:00:00

BMC40770I BUILD STATISTICS: # INPUT RECORDS = 0, # KEY VALUES = 0
Waits for Sort = 0
Getpage calls = 7, Pages written = 5, Leaf pages = 1

Termination complete, elapsed time = 00:00:00
Build phase complete. Accumulated TCB time = 00:00:00

Figure 103: Example 9—AFRSTMT for recovering to a non-DB2 data set

Example 9: Recovering to a non-DB2 data set

RECOVER PLUS FOR DB2 V11.2.00
COPYRIGHT BMC SOFTWARE INC. 1991-2015
RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

PARM LIST:  DB2 SSID = DEDL
UTILID = EXAMPLE10B
UTILITY NEW/RESTART REQUESTED - SYNCPOINT RESTART WILL OCCUR IF UTILID IS IN BMCTIL
RESTART = NEW/RESTART
MSGLEVEL = 2

PARAM Not Coded - CHECKPT
PARAM Not Coded - RDB2STAT
PARAM Not Coded - AFROPTS
PARAM Not Coded - ACAOPTS

----- DEFAULT OPTIONS:
PLANRECV = AFRB111T (MAIN PLAN)
BINDQUALIFIER = BMCAF (QUALIFIER USED FOR DYNAMIC BIND)
PUBLICPLAN = YES (GRANT PLAN TO PUBLIC)
ZIIP = ENABLED (ZIIP PROCESSOR ENABLEMENT)
INDEXLOG = NO (DEFAULT USE OF INDEXLOG STRATEGY)
IXRECP = NO (SET INDEX RECOVER PENDING)
BACKOUT = AUTO (DEFAULT POINT-IN-TIME RECOVERY STRATEGY)

OPND2B2D = YES (ACQUIRE RCF AUTHORITY OF DB2)
RDB2STAT = YES (RESET DB2 OBJECT STATUS)
AMSCAT = NO (USE 'CATALOG' IN IDCAMS INPUT)
WKUNIT = SYSALLDA (WORK UNIT NAME)
WTOR = YES (ISSUE WTOR IF SPACE REMAINS IN STOPP STATUS)

USEHDROBIDS = YES (USE HEADER PAGE IDS FOR OBIDXLAT)
AUX = NO (INCLUDE XML/LOB/ALL/HISTORY/ARCHIVE OBJECTS)
CHECKPT = PHASE (CHECKPOINTING)
CHECKINT = 0 (CHECKPOINT INTERVAL MINUTES)
ERRCONT = 10 (DEFAULT MAX SEVERE ERRORS)
RCLTSK = 10 (MAXIMUM RECALL TASKS)
TBUFFS = 1000 (BUFFER MANAGER PAGES)
HISTORY = YES (UPDATE BMCHIST TABLE)

----- DEFAULT OPTIONS:  SORT
SMCORE = (OK, OK) (SORT CORE VALUES)
RESINV = (OK) (MEMORY BELOW 16MB EXCLUDED FROM USE BY SORT)
SORTDEV = SORT DEFAULT (DEVICE TYPE FOR LOG AND INDEX Sorts)
Example 9: Recovering to a non-DB2 data set
1414560K

BMC40028I UTILID EXAMPLE10B ORIGINALLY TERMINATED BEFORE COMPLETING ANALYZE PHASE. RESTART WILL BE TREATED AS A NEW RUN

BMC40611I INPUT STATEMENTS:
BMC40101I   RECOVER TABLESPACE BMCDBSMP.BMCTS010 INDEP OUTSPACE
BMC40101I   REBUILD INDEX (RMD.BMCIX010) INDEP OUTSPACE NOWORKDDN

BMC40604I ZIIP ENABLED (0) USING SUBSYSTEM XBMA
BMC40937I z/OS RELEASE = 1.13.00, PID = HBB7780, SMS RELEASE = 1.13.0
BMC40157I AVAILABLE REGION BELOW 16M = 9032K, AVAILABLE REGION ABOVE 16M = 1404336K, NUMBER OF CPUS = 5
BMC40020I BYT:0, 0, 0; BLK:0, 0, 0; TOT:3214464, 9766832, 9766832; iter: 1; RC:4

BMC40335I ANALYZE STARTING 08/20/2014 09:51:31

BMC40335I CATALOG LOOKUP STARTING 08/20/2014 09:51:31
BMC40336I PROC RO STMT FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I PROC IX STMT FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSTB FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSIX FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I COL LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSKY FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SCAN SYSFL FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I TS LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I IP LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SG LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40633I USING STANDARD DB2 SECURITY

BMC40336I DB LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I TB LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I IP LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I SG LOOKUP FINISHED.
BMC40287I   ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I   CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

Chapter 5 Examples of NGT Recover jobs 411
Figure 104: Example 9—AFROSUM for recovering to a non-DB2 data set
Example 9: Recovering to a non-DB2 data set

Figure 105: Example 9—AFRPLAN for recovering to a non-DB2 data set

BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2015
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40700I EXECUTION PLAN SUMMARY:

BMC40701I PHASE:  MERGE
BMC40721I PHASE INITIALIZATION:
BMC40301I TABLESPACE = BMCDBSMP.BMCTS010
BMC40738I ALLOCATE INPUT IMAGE COPY FILE
BMC40302I DSN = RDAPXR.BMCDBSMP.BMCTS010.F.T095108
BMC40738I FILE OPTIMIZATION
BMC40301I ALLOCATE 2 INPUT FILE BUFFER(S)
BMC40738I OPEN IMAGE COPY FILE
BMC40302I DSN = RDAPXR.BMCDBSMP.BMCTS010.F.T095108
BMC40738I ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40738I CREATE SPACE
BMC40345I SPACE = BMCDBSMP.BMCTS010
BMC40738I ALLOCATE SPACE (READ/WRITE)
BMC40738I ALLOCATE SPACE (READ/WRITE)
BMC40345I SPACE = BMCDBSMP.BMCTS010
BMC40738I ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40345I SPACE = BMCDBSMP.BMCTS010
BMC40738I FILE OPTIMIZATION
BMC40345I SPACE = BMCDBSMP.BMCTS010
BMC40738I ALLOCATE 2 SPACE OUTPUT BUFFER(S)
BMC40738I ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40702I STEP:  MERGE SPACE
BMC40702I MERGE SPACE
BMC40738I TABLESPACE = BMCDBSMP.BMCTS010
BMC40301I PHASE TERMINATION:
BMC40738I DISCONNECT FROM SPACE
BMC40738I DEALLOCATE SPACE
Figure 106: Example 9—AFRERR for recovering to a non-DB2 data set
Example 10: Overriding installation options

This example demonstrates the use of the OPTIONS statement to override installation options at execution time.

Not all installation options can be overridden. See “OPTIONS command” on page 103 for a description of the OPTIONS statement.

Note
Some OPTIONS keywords are different from their corresponding installation option names. For example, the OPTIONS keyword is MAXDRIVES, but the installation option is MAXDRIVE. Be sure to use the keywords as they are shown in the OPTIONS syntax diagram in “OPTIONS syntax” on page 104.

Example 10A: Overriding several installation options

In the first part of Example 10A, the SORTDEVT and SORTNUM options are overridden to SYSDA and 03.

In addition, the resources eligible for use in this recovery are limited to the archive log 1 and active log 1, and the remote primary and local primary image copies. Note that, in the example output, only an active log was read because it had not yet been archived and was therefore the only source of log records for the recovery.

Figure 107: Example 10A--JCL for overriding installation options

```bash
//AFREX12A JOB (PAFR), 'EXAMPLE 10A',
//         CLASS=Q, NOTIFY=&SYSUID,
//         MSGCLASS=X
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=0M,
//             PARM='DGE, EXAMPLE10A, NEW'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//          DD DISP=SHR, DSN=DB2.DSNEXIT
//          DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
OPTIONS
    SORTDEVT SYSDA SORTNUM 03
RESOURCE SELECTION
    LOGS (ARC1, ACT1)
    COPIES (RP, LP)
RECOVER TABLESPACE BMCDBSMP.BMCTS012
/*
*/
Example 10B: Overriding the IXRECP installation option

The following JCL shows the use of the OPTIONS statement to override the IXRECP installation option.

IXRECP YES is used to set RECP or RBDP for the indexes associated with the recovered table space to force index recoveries before the data can be accessed. NGT Recover also issues a warning message for each index that has not been recovered with the table space.

Figure 108: Example 10B—JCL for overriding the IXRECP installation option

```sql
//AFREX12B JOB (PAFR),'EXAMPLE 10B',
//        CLASS=Q,NOTIFY=&SYSUID,
//        MSGCLASS=X
//*
//RECOVER EXEC PGM=AFRMAIN,REGION=0M,
//             PARM='DGE,EXAMPLE10B,NEW/RESTART,MSGLEVEL(2)'  
//STEPLIB  DD DISP=SHR,DSN=product.libraries
//          DD DISP=SHR,DSN=DB2.DSNEXIT
//          DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN     DD *
OPTIONS IXRECP YES
RECOVER TABLESPACE BMCDBSMP.BMCTS012
TOCOPY LASTCOPY
/*
```

Figure 109 on page 416 through Figure 111 on page 420 show the following output, which is produced by running this JCL:

- AFRSTMT
- AFRPRINT
- AFRPR001
- AFRPLAN

For a description of each of these files, see Table 9 on page 320.

Figure 109: Example 10B—AFRSTMT for overriding the IXRECP installation option
Example 10: Overriding installation options

Chapter 5  Examples of NGT Recover jobs  417
Example 10: Overriding installation options

BMC40967I TAPE DEVICE13 = IBMVTS
BMC40967I TAPE DEVICE14 = PHXATL
BMC40967I TAPE DEVICE15 = SA3590
BMC40967I TAPE DEVICE16 = TAPEABL
BMC40967I TAPE DEVICE17 = 3590B
BMC40967I TAPE DEVICE18 = 3590E
BMC40967I TAPE DEVICE19 = 3480
BMC40967I TAPE DEVICE20 = 3400-9
BMC40967I TAPE DEVICE21 = SYS3480R
BMC40967I TAPE DEVICE22 = SYS348XR

BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40876I MAINT: NO RECOVER PLUS PTFS APPLIED

BMC96010I SOLUTION COMMON CODE V11.2.01
BMC40876I MAINT: NO SOLUTION COMMON CODE PTFS APPLIED

BMC96010I BMCSORT ENGINE V2.3.01
BMC40876I MAINT: BPJ0195 BPJ0198 BPJ0209 BPJ0250 BPJ0263 BPJ0269 BPJ0279
BMC40876I MAINT: BPJ0288 BPJ0308 BPJ0343 BPJ0357 BPJ0361 BPJ0374 BPJ0390
BMC40876I MAINT: BPJ0404 BPJ0473 BPJ0503 BPJ0521

BMC40335I PACKAGE CHECKING AND BINDING STARTING 08/21/2014 17:15:06
BMC40336I PACKAGE CHECKING AND BINDING FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:01
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40020I EBCDIC CCSID = 37
BMC40954I AVAILABLE MEMORY BELOW THE LINE = 10068K, TOTAL AVAILABLE MEMORY (AT LEAST) = 1373592K

BMC40633I USING STANDARD DB2 SECURITY
Figure 110: Example 10B—AFRPLAN for overriding the IXRECP installation option

Example 10: Overriding installation options
BMC40700I EXECUTION PLAN SUMMARY:

BMC40701I PHASE: MERGE
BMC40721I PHASE INITIALIZATION:
BMC40301I TABLESPACE = BMCDBSMP.BMCTS012
BMC40738I ALLOCATE INPUT IMAGE COPY FILE
BMC40302I DSN = RDAPXR.BMCDBSMP.BMCTS012.F.T135427.LP
BMC40738I FILE OPTIMIZATION
BMC40738I ALLOCATE 2 INPUT FILE BUFFER(S)
BMC40738I OPEN IMAGE COPY FILE
BMC40738I DSN = RDAPXR.BMCDBSMP.BMCTS012.F.T135427.LP
BMC40738I ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I SET RECOVERY PENDING (RECP) FLAG
BMC40302I DSN = RDAPXR.BMCDBSMP.BMCTS012.F.T135427.LP
BMC40738I CREATE SPACE
BMC40345I SPACE = BMCDBSMP.BMCTS012
BMC40738I ALLOCATE SPACE (READ/WRITE)
BMC40345I SPACE = BMCDBSMP.BMCTS012
BMC40738I CONNECT TO SPACE
BMC40738I ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I FILE OPTIMIZATION
BMC40738I ALLOCATE 2 SPACE OUTPUT BUFFER(S)
BMC40738I ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40702I STEP: MERGE SPACE
BMC40738I MERGE SPACE
BMC40301I TABLESPACE = BMCDBSMP.BMCTS012
BMC40735I PHASE TERMINATION:
BMC40738I DISCONNECT FROM SPACE
BMC40738I DEALLOCATE SPACE
BMC40345I SPACE = BMCDBSMP.BMCTS012
BMC40738I FREE PAGE CHECK BUFFER
BMC40738I FREE SPACEMAP BUFFER
BMC40738I FREE PASTE BUFFER
BMC40738I FREE 2 OUTPUT SPACE BUFFER(S)
BMC40738I FREE PAGE CHECK BUFFER
BMC40738I FREE PASTE BUFFER
BMC40738I FREE SPACEMAP BUFFER
BMC40738I FREE 2 OUTPUT SPACE BUFFER(S)
BMC40738I FREE PAGE CHECK BUFFER
BMC40738I FREE PASTE BUFFER
BMC40738I FREE SPACEMAP BUFFER
BMC40738I CLOSE IMAGE COPY FILE
BMC40302I DSN = RDAPXR.BMCDBSMP.BMCTS012.F.T135427.LP
BMC40738I FREE 2 INPUT FILE BUFFER(S)
BMC40738I DEALLOCATE INPUT IMAGE COPY FILE
BMC40302I DSN = RDAPXR.BMCDBSMP.BMCTS012.F.T135427.LP
BMC40738I RESET RECOVERY PENDING (RECP) FLAG
BMC40738I REGISTER PARTIAL RECOVERY IN SYSCOPY
BMC40738I SET INDEX(S) STATUS TO REBUILD PENDING IF NEEDED
BMC40738I RECORD RESTART INFORMATION

Figure 111: Example 10B—AFRPRINT and AFRPR001 for overriding the IXRECP installation option
Example 10: Overriding installation options

Chapter 5 Examples of NGT Recover jobs 421
Example 11: Recovering using a change accumulation file

This example illustrates the creation and use of change accumulation files in the recovery of a table space. An index on the table in the table space is also rebuilt.

Example 11A: Creating the change accumulation file

The JCL in the following figure creates the change accumulation file.

Figure 112: Example 11A—JCL using R+/CHANGE ACCUM to accumulate log records

```bash
//AFREX13A JOB (PAFR), 'EXAMPLE 11A',
//          CLASS=Q, NOTIFY=&SYSUID,
//          MSGCLASS=X
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=0M,
//          PARM='DEBA, EXAMPLE11A, NEW/RESTART, MSGLEVEL(2)'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//          DD DISP=SHR, DSN=DB2.DSNEXIT
//          DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
ACCUM GROUP BMCSMPGP
/*

Example 11B: Using the change accumulation file

The JCL in the following figure recovers the table space using the change accumulation file created in Example 11A.

See “Example 11A: Creating the change accumulation file” on page 422.

Figure 113: Example 11B--JCL recovering a table space using a change accumulation file

```bash
//AFREX13B JOB (PAFR), 'EXAMPLE 11B',
//          CLASS=Q, NOTIFY=&SYSUID,
//          MSGCLASS=X
/*ROUTE XEQ BMCPLX1
/*JOBPARM SYSAFF=DB2A
//*
//RECOVER EXEC PGM=AFRMAIN, REGION=0M,
//          PARM='DEBA, EXAMPLE11B, NEW/RESTART, MSGLEVEL(2)'
//STEPLIB DD DISP=SHR, DSN=product.libraries
//          DD DISP=SHR, DSN=DB2.DSNEXIT
//          DD DISP=SHR, DSN=DB2.DSNLOAD
//SYSIN DD *
OPTIONS USEACCUM YES
RECOVER TABLESPACE BMCDBSMP.BMCTS020
Example 12: Using the MAINT parameter

Example 12 illustrates the use of the MAINT parameter on the EXEC statement to print a record of applied maintenance to AFRSUMRY and AFRSTMT. Common tables used by BMC utilities are also printed to AFRSTMT.

Figure 114: Example 12--JCL using the MAINT parameter

Example 13: Using the LOGSCAN command

Example 13 illustrates the use of the LOGSCAN command. LOGSCAN gathers information about log records that you can use to determine your recovery strategy.

The LOGSCAN command uses the SYSSCAN DD statement included in this example to designate output for the report that the LOGSCAN command generates.

Figure 115: Example 13--JCL using the LOGSCAN command
The SYSSCAN output produced by running this JCL is shown in the following figure. AFRPRINT and AFRPR001 are shown in Figure 117 on page 424. See Table 9 on page 320 for a description of each of these files.

Figure 116: Example 13--SYSSCAN for the LOGSCAN command

<table>
<thead>
<tr>
<th>Message</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40018I</td>
<td>RECOVER PLUS FOR DB2 V11.2.00</td>
</tr>
<tr>
<td>BMC40860I</td>
<td>LOG SORT STATISTICS (LOGSCAN)</td>
</tr>
<tr>
<td>BMC40861I</td>
<td>08/21/2014 17:45:41</td>
</tr>
<tr>
<td>BMC40512I</td>
<td>LOG FILE RESOURCES:</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = DSNDXW.DXW1.LOGCOPY1.DS02.DATA</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = DSNDXW.DXW3.LOGCOPY1.DS02.DATA</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = DSNDXW.DXW4.LOGCOPY1.DS03.DATA</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = DSNDXW.DXW2.LOGCOPY1.DS06.DATA</td>
</tr>
<tr>
<td>BMC40566I</td>
<td>USING LOG DSNDXW.DXW1.LOGCOPY1.DS02.DATA FOR PROCESSING</td>
</tr>
<tr>
<td>BMC40555I</td>
<td>FROMRBA = X'0000000000222C08A000' TORBA = X'0000000000222C86FFFF'</td>
</tr>
<tr>
<td>BMC40566I</td>
<td>USING LOG DSNDXW.DXW3.LOGCOPY1.DS02.DATA FOR PROCESSING</td>
</tr>
<tr>
<td>BMC40555I</td>
<td>FROMRBA = X'00000000001877CA3000' TORBA = X'00000000001877BC6AFF'</td>
</tr>
<tr>
<td>BMC40566I</td>
<td>USING LOG DSNDXW.DXW4.LOGCOPY1.DS03.DATA FOR PROCESSING</td>
</tr>
<tr>
<td>BMC40555I</td>
<td>FROMRBA = X'000000000015BA0F0D000' TORBA = X'000000000015BABED4FF'</td>
</tr>
<tr>
<td>BMC40566I</td>
<td>USING LOG DSNDXW.DXW2.LOGCOPY1.DS06.DATA FOR PROCESSING</td>
</tr>
<tr>
<td>BMC40555I</td>
<td>FROMRBA = X'0000000000191E5FB000' TORBA = X'000000000019DFBBAFF'</td>
</tr>
<tr>
<td>BMC40189I</td>
<td>THE 0 TAPE AND 4 DISK LOG FILES WILL BE READ IN 2 LOG READING STEPS BECAUSE MAXDRIVES = 0 AND MAXLOGS = 3</td>
</tr>
<tr>
<td>BMC40850I</td>
<td>OBJECT</td>
</tr>
<tr>
<td>BMC40851I</td>
<td>BMCDBSMP.BMCTS014</td>
</tr>
<tr>
<td>BMC40851I</td>
<td>BMCDBSMP.BMCIX014</td>
</tr>
<tr>
<td>BMC40852I</td>
<td>TOTAL</td>
</tr>
<tr>
<td>BMC40853I</td>
<td>NUMREC EST 0 AVGRECSZ 0</td>
</tr>
</tbody>
</table>

Figure 117: Example 13--AFRPRINT and AFRPR001

<table>
<thead>
<tr>
<th>Message</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC40018I</td>
<td>RECOVER PLUS FOR DB2 V11.2.00</td>
</tr>
<tr>
<td>BMC40473I</td>
<td>COPYRIGHT BMC SOFTWARE INC. 1991-2015</td>
</tr>
<tr>
<td>BMC96173I</td>
<td>RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001</td>
</tr>
<tr>
<td>BMC40474I</td>
<td>CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO <a href="mailto:SUPPORT@BMC.COM">SUPPORT@BMC.COM</a></td>
</tr>
<tr>
<td>BMC40018I</td>
<td>RECOVER PLUS FOR DB2 V11.2.00</td>
</tr>
<tr>
<td>BMC40473I</td>
<td>COPYRIGHT BMC SOFTWARE INC. 1991-2015</td>
</tr>
<tr>
<td>BMC96173I</td>
<td>RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001</td>
</tr>
<tr>
<td>BMC40474I</td>
<td>CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO <a href="mailto:SUPPORT@BMC.COM">SUPPORT@BMC.COM</a></td>
</tr>
<tr>
<td>BMC40016I</td>
<td>LOGSCAN PHASE STARTING 08/21/2014 17:45:44</td>
</tr>
<tr>
<td>BMC40725I</td>
<td>PROCESSING LOG RANGE:</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = DSNDXW.DXW2.LOGCOPY1.DS06.DATA</td>
</tr>
<tr>
<td>BMC40555I</td>
<td>FROMRBA = X'00000000001C91E5FB000' TORBA = X'00000000001C9DFBBAFF'</td>
</tr>
<tr>
<td>BMC40724I</td>
<td>DATA SHARING MEMBERID IS 4</td>
</tr>
<tr>
<td>BMC40725I</td>
<td>PROCESSING LOG RANGE:</td>
</tr>
<tr>
<td>BMC40302I</td>
<td>DSN = DSNDXW.DXW4.LOGCOPY1.DS03.DATA</td>
</tr>
<tr>
<td>BMC40555I</td>
<td>FROMRBA = X'000000000015BA0F0D000' TORBA = X'000000000015BABED4FF'</td>
</tr>
<tr>
<td>BMC40724I</td>
<td>DATA SHARING MEMBERID IS 3</td>
</tr>
<tr>
<td>BMC40725I</td>
<td>PROCESSING LOG RANGE:</td>
</tr>
</tbody>
</table>
Example 13: Using the LOGSCAN command

Chapter 5  Examples of NGT Recover jobs  425
Example 14: Simulating recovery

Example 14 illustrates the use of the SIMULATE YES option to invoke recovery simulation.

Specifying SIMULATE YES necessitates the use of SIMRCVR TABLESPACE and SIMRBLD INDEX (instead of RECOVER TABLESPACE and REBUILD INDEX).

**Note**
Simulation is a feature of the BMC Recovery Management for DB2 solution and requires a valid Recovery Management solution password.

**Example 14: Simulating recovery**

Example 14 illustrates the use of the SIMULATE YES option to invoke recovery simulation.

Specifying SIMULATE YES necessitates the use of SIMRCVR TABLESPACE and SIMRBLD INDEX (instead of RECOVER TABLESPACE and REBUILD INDEX).

**Note**
Simulation is a feature of the BMC Recovery Management for DB2 solution and requires a valid Recovery Management solution password.

**Figure 118: Example 14—JCL using the SIMULATE option**

```
//AFREX18  JOB (PAFR), 'EXAMPLE 14',
//         CLASS=Q, NOTIFY=&SYSUID,
//         MSGCLASS=X
/*ROUTE XEQ BMCPLX1
/*JOBPARM SYSAFF=DB2A
/**
/*RECOVER EXEC PGM=AFRMAIN, REGION=0M,
/*     PARM='DGE, EXAMPLE14, NEW/RESTART, MSGLEVEL(2)'
/*STEPLIB DD DISP=SHR, DSN=product.libraries
/*     DD DISP=SHR, DSN=DB2.DSNEXIT
/*     DD DISP=SHR, DSN=DB2.DSNLOAD
/*SYSIN DD *
OPTIONS SIMULATE YES
SIMRCVR TABLESPACE BMCDBSMP.BMCTS014
SIMRBLD INDEX (ALL) TABLESPACE BMCDBSMP.BMCTS014
/*
/*
```

Example 15: Using MAXKSORT for parallel index rebuilds

Example 15 illustrates the use of the MAXKSORT option to invoke parallel index sorts and rebuilds to rebuild a partitioned index and multiple nonpartitioned indexes.

**Figure 119: Example 15—JCL using the MAXKSORT option for parallel index rebuilds**

```
//AFREX16  JOB (PAFR), 'EXAMPLE 15',
//         CLASS=Q, NOTIFY=&SYSUID,
//         MSGCLASS=X, TIME=(0,5)
/*ROUTE XEQ BMCPLX1
/*JOBPARM SYSAFF=DB2A
/**
/*RECOVER EXEC PGM=AFRMAIN, REGION=0M,
/*     PARM='DEBA, EXAMPLE15, NEW/RESTART, MSGLEVEL(0)'
/*STEPLIB DD DISP=SHR, DSN=product.libraries
/*     DD DISP=SHR, DSN=DB2.DSNEXIT
```
Figure 121 on page 428 through Figure 123 on page 431 show the following output, which is produced by running this JCL:

- AFRERR
- AFRSUMRY
- AFRSTMT
- AFRPRINT
- AFRPR001 (AFRPR002 through AFRPR009 are produced by not shown.

For a description of each of these files, see Table 9 on page 320.
Example 15: Using MAXKSORT for parallel index rebuilds

Figure 121: Example 15—AFRERR for parallel index rebuilds with MAXKSORT

```
BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2015
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM
BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2015
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM
BMC40775W THERE ARE NO KEY VALUES FOR PART 2 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 3 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 4 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 5 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 6 OF INDEX RDAPXR.IX67P32
```

Figure 122: Example 15—AFRSTMT for parallel index rebuilds with MAXKSORT

```
BMC40775W THERE ARE NO KEY VALUES FOR PART 2 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 3 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 4 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 5 OF INDEX RDAPXR.IX67P32
BMC40775W THERE ARE NO KEY VALUES FOR PART 6 OF INDEX RDAPXR.IX67P32
```
Example 15: Using MAXKSORT for parallel index rebuilds

Chapter 5  Examples of NGT Recover jobs  429
Example 15: Using MAXKSORT for parallel index rebuilds

BMC40037I UTILID EXAMPLE16 WAS NOT FOUND IN THE BMCUTIL TABLE - UTILITY WILL RUN AS IF 'NEW' WERE SPECIFIED

BMC40611I INPUT STATEMENTS:
BMC40101I OPTION MAXKSORT 12
BMC40101I REBUILD INDEX (ALL) TABLESPACE RMDSUPDB.TS67P32K
BMC40101I
BMC40957I MAXKSORT VALUE OVERRIDDEN, NEW VALUE = 12
BMC96004I ZIIP ENABLED (0) USING SUBSYSTEM XBMA
BMC40157I AVAILABLE REGION BELOW 16M = 9032K, AVAILABLE REGION ABOVE 16M = 1404328K, NUMBER OF CPUS = 5
BMC40020I BYT:0, 0, 0; BLK:0, 0, 0; TOT:1656364, 10037136, 10037136; iter: 1; RC:4
BMC40335I ANALYZE STARTING 08/21/2014 18:05:06
BMC40335I CATALOG LOOKUP STARTING 08/21/2014 18:05:06
BMC40336I PROC IX STMT FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40336I SCAN SYSB FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40336I SCAN SYSX FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40336I COL LOOKUP FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40336I SCAN SYSKY FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40336I SCAN SYSFL FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40681I THE KEYS FOR INDEX RDAPXR.IX67P324 WILL GO DIRECTLY TO THE SORT BECAUSE THERE IS NO SYSUT1 DD STATEMENT IN THE JCL
BMC40681I THE KEYS FOR INDEX RDAPXR.IX67P323 WILL GO DIRECTLY TO THE SORT BECAUSE THERE IS NO SYSUT1 DD STATEMENT IN THE JCL
BMC40681I THE KEYS FOR INDEX RDAPXR.IX67P322 WILL GO DIRECTLY TO THE SORT BECAUSE THERE IS NO SYSUT1 DD STATEMENT IN THE JCL
BMC40681I THE KEYS FOR INDEX RDAPXR.IX67P32 WILL GO DIRECTLY TO THE SORT BECAUSE THERE IS NO SYSUT1 DD STATEMENT IN THE JCL
BMC40336I TS LOOKUP FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40633I USING STANDARD DB2 SECURITY
BMC40336I DB LOOKUP FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40336I TB LOOKUP FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
BMC40336I IP LOOKUP FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:00
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
Example 15: Using MAXKSORT for parallel index rebuilds

Figure 123: Example 15—AFRPRINT for parallel index rebuilds with MAXKSORT
In addition to AFRPR001 in the following figure, this job also produces AFRPR002 through AFRPR009.

### Figure 124: Example 15—AFRPR001 for parallel index rebuilds with MAXKSORT
Example 15: Using MAXKSORT for parallel index rebuilds

BMC40593I START UNLOAD COMPLETE, ELAPSED TIME = 00:00:00
BMC40102I UNLOAD KEYS STEP STARTING
BMC403501 DATA SET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00008 ELAPSED = 00:00:00
BMC40593I INITIALIZATION COMPLETE, ELAPSED TIME = 00:00:00
BMC96303I KEY SORT FOR INDEX GROUP 1 WILL USE THREAD 1
BMC40954I AVAILABLE MEMORY BELOW THE LINE = 8772K, TOTAL AVAILABLE MEMORY (AT LEAST) = 1402096K
BMC40441I BMCSORT PRODUCT LOCATED AND WILL BE USED
BMC40020I ORG-BEL:0, 0, 0; EXT:0, 0, 0; TOT:1643692, 9993468, 9993468; ITER: 1; RC:4
BMC40906I NUMBER OF PAGES EXAMINED = 11112, ROWS EXAMINED = 300000, KEYS EXTRACTED = 1200000
BMC40907I INDIRECT REFERENCES: FORWARD = 0, BACKWARD = 0
BMC40901I TOTAL WAITS = 92
BMC40916I TABLESPACE READ WAITS = 53, KEY OUT WAITS = 39
BMC40917I TOTAL TIME FOR WAITS = 00:00:01
BMC40918I TABLESPACE READ WAIT TIME = 00:00:01, KEY OUT WAIT TIME = 00:00:00
BMC40952I NUMBER OF PAGES FROM INPUT SPACE = 11115
BMC40900I UNLOAD KEYS STEP COMPLETE
BMC40350I DATA SET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001 WAS SUCCESSFULLY UNALLOCATED
BMC40593I UNLOAD KEY PROCESS COMPLETE, ELAPSED TIME = 00:00:02
BMC96303I KEY SORT FOR INDEX GROUP 2 WILL USE THREAD 2
BMC40954I AVAILABLE MEMORY BELOW THE LINE = 8692K, TOTAL AVAILABLE MEMORY (AT LEAST) = 1399576K
BMC40441I BMCSORT PRODUCT LOCATED AND WILL BE USED
BMC40020I ORG-BEL:0, 0, 0; EXT:0, 0, 0; TOT:1640852, 9990628, 9990628; ITER: 1; RC:4
BMC40906I NUMBER OF PAGES EXAMINED = 11112, ROWS EXAMINED = 914053
BMC40907I INDIRECT REFERENCES: FORWARD = 0, BACKWARD = 0
BMC40901I TOTAL WAITS = 30
BMC40916I TABLESPACE READ WAITS = 3; KEY OUT WAITS = 262144
BMC40917I TOTAL TIME FOR WAITS = 00:00:01
BMC40918I TABLESPACE READ WAIT TIME = 00:00:01, KEY OUT WAIT TIME = 00:00:00
BMC40952I NUMBER OF PAGES FROM INPUT SPACE = 11115
BMC40900I UNLOAD KEYS STEP COMPLETE
BMC40350I DATA SET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001 WAS SUCCESSFULLY UNALLOCATED
BMC40016I BUILD PHASE STARTING 08/21/2014 18:05:12
BMC40303I INDEX = RDAPXR.IX67P32
BMC40305I DSNUM = 1
BMC40484I MESSAGES FROM PHASE SUBTASK:
BMC40060I PENDING FLAG RBDP SET FOR
BMC40346I INDEXSPACE = RMDSUPDB.IX67P32
BMC40305I DSNUM = 1
BMC40356I AN IDCAMS 'DELETE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001
BMC40356I AN IDCAMS 'DEFINE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001
BMC40350I DATA SET DEDLCAT.DSNDBD.RMDSUPDB.IX67P32.I0001.A001 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00015 ELAPSED = 00:00:00
BMC40016I BUILD PHASE COMPLETE 08/21/2014 18:06:24
BMC40303I INDEX = RDAPXR.IX67P32
BMC40305I DSNUM = 1
BMC40484I END OF PHASE SUBTASK MESSAGES
BMC40016I BUILD PHASE COMPLETE 08/21/2014 18:06:24
BMC40601I PENDING FLAG RBDP SET FOR
BMC40346I INDEXSPACE = RMDSUPDB.IX67P32
BMC40305I DSNUM = 1
BMC40356I AN IDCAMS 'DELETE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001
BMC40356I AN IDCAMS 'DEFINE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001
BMC40350I DATA SET DEDLCAT.DSNDBD.RMDSUPDB.IX67P32.I0001.A001 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00015 ELAPSED = 00:00:00
BMC40016I BUILD PHASE COMPLETE 08/21/2014 18:06:24
BMC40601I PENDING FLAG RBDP SET FOR
BMC40346I INDEXSPACE = RMDSUPDB.IX67P32
BMC40305I DSNUM = 1
BMC40356I AN IDCAMS 'DELETE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001
BMC40356I AN IDCAMS 'DEFINE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBD.RMDSUPDB.TS67P32K.I0001.A001
BMC40350I DATA SET DEDLCAT.DSNDBD.RMDSUPDB.IX67P32.I0001.A001 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00015 ELAPSED = 00:00:00
BMC40016I BUILD PHASE COMPLETE 08/21/2014 18:06:24
Example 16: Using MAXKSORT and recovering a table space

Example 16 illustrates the recovery of a table space and the use of the MAXKSORT option to invoke parallel index sorts and rebuilds for the partitioned index and the nonpartitioned indexes.

Figure 125: Example 16--JCL for table space recovery and index recovery using the MAXKSORT option

```
//AFREX17  JOB (PAFR),'EXAMPLE 16',
  //         CLASS=Q,NOTIFY=&SYSUID,
  //         MSGCLASS=X
```

Example 16: Using MAXKSORT and recovering a table space

Example 16 illustrates the recovery of a table space and the use of the MAXKSORT option to invoke parallel index sorts and rebuilds for the partitioned index and the nonpartitioned indexes.

Figure 125: Example 16--JCL for table space recovery and index recovery using the MAXKSORT option

```
//AFREX17  JOB (PAFR),'EXAMPLE 16',
  //         CLASS=Q,NOTIFY=&SYSUID,
  //         MSGCLASS=X
```
Example 17: Using timestamp recovery

The inflight resolution technology of the BMC Recovery Management for DB2 solution enables you to perform a consistent recovery to any timestamp or LRSN/RBA.

The ability to resolve inflight units of work at any point in time or to any log point completely eliminates the need to perform quiesces to establish consistent recovery points during application execution. The ability to avoid quiesces can dramatically improve the availability of your DB2 data.

The timestamp recovery feature of BMC Recovery Management for DB2 solution uses inflight resolution technology to perform a consistent point-in-time recovery to any user-specified timestamp. The Recovery Management solution translates the timestamp to an RBA or LRSN, recovers the objects, then resolves all inflight units of work for both data sharing and non-data-sharing systems. (The online interface of RECOVERY MANAGER for DB2 supports this feature on data sharing systems.)

Note

Timestamp recovery and inflight resolution technology use features from the RECOVERY MANAGER for DB2, BMC Next Generation Technology Recover for DB2 for z/OS, and Log Master for DB2 components of the BMC Recovery Management for DB2 solution. The Recovery Management solution password is required to use the feature.

For more information, see the Recovery Management for DB2 User Guide and the RECOVERY MANAGER for DB2 User Guide.

The following figure is an example of NGT Recover JCL that includes the options for a timestamp recovery.

Figure 126: Example 17—JCL using RECOVERYPOINT for timestamp recovery

//AFREX18 JOB (PAFR),’EXAMPLE 17’, // CLASS=Q,NOTIFY=&SYSUID,
Example 18: Using the MAXLSORT default value

In this example, NGT Recover does a basic recovery of table space and uses the default value of the MAXLSORT installation option so that NGT Recover uses concurrent log sorts. If you need to override the default value for MAXLSORT, you can specify a MAXLSORT value on the OPTIONS command in the SYSIN.

If indexes were recovered in the job, the default value of the KSORTSHARE installation option (YES) would specify that NGT Recover should spread the key sorts over all tasks. You can also specify KSORTSHARE on the OPTION command.

Figure 127: Example 18—JCL using default values

Figure 128 on page 437 through Figure 135 on page 447 show the following output, which is produced by running this JCL:

- AFRSTMT
- AFRSUMRY
- AFRPRINT
- AFRPR001
- AFRPR002
AFRPR003
AFROSUM
AFRPLAN

For a description of each of these files, see “DD statements common to all NGT Recover executions” on page 319.

The AFRSTMT output shown in the following figure shows the MAXLSORT value as 50 for this job. The MAXLSORT installation option defaults to 0, which indicates that NGT Recover will determine the value of MAXLSORT for a job. KSORTSHARE and MAXKSORT are also highlighted in this output.

Figure 128: Example 18—AFRSTMT showing the MAXLSORT value

```
BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2015
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40931I PARM LIST:  DB2 SSID = DEDL
BMC40932I            UTILID = RECVR19
BMC40008I UTILITY REPLACEMENT ('NEW') REQUESTED
BMC40931I             RESTART = NEW
BMC40934I            MSGLEVEL = 2
BMC40930I PARM NOT CODED - CHECKPT
BMC40930I PARM NOT CODED - RDB2STAT
BMC40930I PARM NOT CODED - AFROPTS
BMC40930I PARM NOT CODED - ACAOPTS
BMC96035I ----- DEFAULT OPTIONS:
BMC40924I            PLANRECV = AFRB111T       (MAIN PLAN)
BMC40408I         BINDQUALIFIER = BMCAF         (QUALIFIER USED FOR DYNAMIC BIND)
BMC40241I            PUBLICPLAN = YES            (GRANT PLAN TO PUBLIC)
BMC96003I                  ZIIP = ENABLED        (ZIIP PROCESSOR ENABLEMENT)
BMC40547I              INDEXLOG = NO             (DEFAULT USE OF INDEXLOG STRATEGY)
BMC40941I            IXRECP = NO             (SET INDEX RECOVER PENDING)
BMC96233I            BACKOUT = AUTO            (DEFAULT POINT-IN-TIME RECOVERY STRATEGY)
BMC40944I            OPNDB2ID = YES            (ACQUIRE RACF AUTHORITY OF DB2)
BMC40927I            RDB2STAT = YES            (RESET DB2 OBJECT STATUS)
BMC40946I            AMSCAT = NO             (USE 'CATALOG' IN IDCAMS INPUT)
BMC40938I            WKUNIT = SYSALLDA       (WORK UNIT NAME)
BMC40878I            WTOR = YES            (ISSUE WTOR IF SPACE REMAINS IN STOPP STATUS)
BMC96034I            USEHDROBIDS = YES            (USE HEADER PAGE IDS FOR OBIDXLAT)
BMC96235I            AUX = NO            (INCLUDE XML/LOB/ALL/HISTORY/ARCHIVE OBJECTS)
BMC40926I            CHECKPT = PHASE          (CHECKPOINTING)
BMC40945I            CHECKINT = 0            (CHECKPOINT INTERVAL MINUTES)
BMC40537I            ERRCONT = 10            (DEFAULT MAX SEVERE ERRORS)
BMC40956I            RCLTSK = 10            (MAXIMUM RECALL TASKS)
BMC40925I            TRUFFS = 1000           (BUFFER MANAGER PAGES)
BMC96245I            HISTORY = YES            (UPDATE BMCHIST TABLE)
BMC96035I ----- DEFAULT OPTIONS: SORT
BMC40928I            SMCORE = ( OK, OK) (SORT CORE VALUES)
BMC40893I            RESINV = ( OK) (MEMORY BELOW 16MB EXCLUDED FROM USE BY SORT)
BMC40962I            SORTDEV = SORT DEFAULT (DEVICE TYPE FOR LOG AND INDEX SORTS)
BMC409631            SORTNUM = SORT DEFAULT (WORK DATASETS FOR LOG AND INDEX SORTS)
BMC96230I            KSORTSHARE = YES            (KEY SORTS SHARED BY ALL TASKS)
```
Example 18: Using the MAXLSORT default value

**MAXLSORT = 10 (DEFAULT) (MAX CONCURRENT INDEX KEY SORTS)**

**MAXDRIVE = 0**  (DEFAULT MAX TAPE DRIVES)

**MAXLOGS = 3**  (DEFAULT MAX CONCURRENT LOG FILES)

**OUTCOPY = ASCODED**  (OUTCOPY CREATION DIRECTIVE)

**AUTOSIZE = YES**  (AUTO SIZE OUTPUT COPIES AND OUTPUT ACCUM FILES)

**EATTR = NONE**  (EXTENDED ATTRIBUTES FOR DYNAMIC ALLOCATION OF OUTPUT COPIES)

**DISKIORATE = 100**  (DISK READ RATE IN MB/SEC)

**CPUMIPS = 200**  (CPU PROCESSING SPEED IN MIPS)

**COPIES = (FC,LP,LB)**  (DEFAULT SEQUENCE FOR IMAGE COPIES)

**ACCUMS = (LP,LB)**  (DEFAULT SEQUENCE FOR CA FILES)

**LOGS = (ACT1,ACT2 ARC1,ARC2)**  (DEFAULT SEQUENCE FOR LOG FILES)

**TAPE DEVICES**

**TAPE DEVICE01 = 3590-1**

**TAPE DEVICE02 = 3490**

**TAPE DEVICE03 = 3480X**

**TAPE DEVICE04 = ATLVT5**

**TAPE DEVICE05 = ATL3592**

**TAPE DEVICE06 = CART**

**TAPE DEVICE07 = CARTXV**

**TAPE DEVICE08 = CTBTNS**

**TAPE DEVICE09 = CRTNS**

**TAPE DEVICE10 = EXT3592**

**TAPE DEVICE11 = IBMATL**

**TAPE DEVICE12 = IBMVTS**

**TAPE DEVICE13 = PHXATL**

**TAPE DEVICE14 = SA3590**

**TAPE DEVICE15 = TAPEABL**

**TAPE DEVICE16 = 3590B**

**TAPE DEVICE17 = 3590E**

**TAPE DEVICE18 = 3480**

**TAPE DEVICE19 = 3400-9**

**TAPE DEVICE20 = SYS3480R**

**TAPE DEVICE21 = SYS348XR**

**RECOVER PLUS FOR DB2 V11.2.00**

**SOLUTION COMMON CODE V11.2.01**

**BMCSORT ENGINE V2.3.01**

**EBCDIC CCSID = 37**

**DB2 VERSION = 910 SITE TYPE = LOCAL MODE = NFM**

**AVAILABLE MEMORY BELOW THE LINE = 9044K, TOTAL AVAILABLE MEMORY (AT LEAST) = 1414556K**
Example 18: Using the MAXLSORT default value

Chapter 5  Examples of NGT Recover jobs    439
Example 18: Using the MAXLSORT default value

BMC40335I SYNCHRONIZATION STARTING 08/22/2014 17:16:43
BMC40025I OUTPUT FROM COMMAND '-STO DB(G4GLM0DB) SPACE(V9LMQTS1) AT (COMMIT)' FOLLOWS:
BMC40020I DSN9022I *DEDL DSNTDIS 'STOP DATABASE' NORMAL COMPLETION
BMC40336I SYNCHRONIZATION FINISHED.
BMC40287I ELAPSED TIME = 00:00:02, UTILITY ELAPSED TIME = 00:00:03
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC96306I MAXLSORT REDUCED FROM 50 TO 23 DUE TO MEMORY CONSTRAINTS
BMC40336I DDSCAN FINISHED.
BMC40287I ELAPSED TIME = 00:00:00, UTILITY ELAPSED TIME = 00:00:03
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40335I PLANNING STARTING 08/22/2014 17:16:45
BMC40459I BSDS DEDLCAT.BSDS01 SELECTED
BMC40066I TIME TO READ SYSLGRNX WAS 00:00:00
BMC40459I BSDS DEDLCAT.BSDS01 SELECTED

BMC40335I PLAN EXECUTION STARTING 08/22/2014 17:16:49
BMC40025I OUTPUT FROM COMMAND '-STA DB(G4GLM0DB) SPACE(V9LMQTS1) PART(1) ACCESS(RO)' FOLLOWS:
BMC40020I DSN9022I *DEDL DSNTDDIS 'START DATABASE' NORMAL COMPLETION
BMC40025I OUTPUT FROM COMMAND '-STA DB(G4GLM0DB) SPACE(V9LMQTS1) PART(2) ACCESS(RO)' FOLLOWS:
BMC40020I DSN9022I *DEDL DSNTDDIS 'START DATABASE' NORMAL COMPLETION
BMC40025I OUTPUT FROM COMMAND '-STA DB(G4GLM0DB) SPACE(V9LMQTS1) PART(3) ACCESS(RO)' FOLLOWS:
BMC40020I DSN9022I *DEDL DSNTDDIS 'START DATABASE' NORMAL COMPLETION

Figure 129: Example 18--AFRSUMRY

BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40473I COPYRIGHT BMC SOFTWARE INC. 1991-2015
BMC96173I RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001
BMC40474I CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

BMC40001I UTILITY EXECUTION STARTING 08/22/2014 17:16:42
BMC40018I RECOVER PLUS FOR DB2 V11.2.00
BMC40876I MAINT:  NO RECOVER PLUS PTFS APPLIED
BMC96010I SOLUTION COMMON CODE V11.2.01
BMC40876I MAINT:  NO SOLUTION COMMON CODE PTFS APPLIED
BMC96010I BMCSORT ENGINE V2.3.01
BMC40876I MAINT:  BPJ0195 BPJ0198 BPJ0209 BPJ0250 BPJ0263 BPJ0269 BPJ0279
BMC40876I MAINT:  BPJ0288 BPJ0308 BPJ0343 BPJ0357 BPJ0361 BPJ0374 BPJ0390
BMC40876I MAINT:  BPJ0404 BPJ0473 BPJ0503 BPJ0521

BMC40002I UTILITY ID = RECVR19. DB2 SUBSYSTEM ID = DEDL.
BMC40094I ATTEMPTING TO CONNECT TO DB2 SUBSYSTEM DEDL, USING PLAN AFRB111T
BMC40024I SUCCESSFUL CONNECT TO DEDL (RELEASE 910 OF DB2) USING PLAN AFRB111T

BMC40475I INPUT STATEMENTS PRINTED IN AFRSTMFT
BMC40336I ANALYZE FINISHED.
BMC40287I ELAPSED TIME = 00:00:02, UTILITY ELAPSED TIME = 00:00:03
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00

BMC40336I PLANNING FINISHED.
BMC40287I ELAPSED TIME = 00:00:03, UTILITY ELAPSED TIME = 00:00:06
BMC40288I CPU TIME = 00:00:00, UTILITY CPU TIME = 00:00:00
The BMC96400I messages highlighted in AFRPRINT in the following figure point to the AFRPRxxx output files. The BMC96304I messages highlighted in the AFRPR001, AFRPR002, and AFRPR003 in Figure 131 on page 441 through Figure 133 on page 444 indicate the use of parallel merges.

**Figure 130: Example 18--AFRPRINT**

BMC96400I MERGE PHASE MESSAGES FOR THE FOLLOWING OBJECT ARE IN FILE AFRPR001
BMC40301I TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I DSNUM = 1

BMC96400I MERGE PHASE MESSAGES FOR THE FOLLOWING OBJECT ARE IN FILE AFRPR002
BMC40301I TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I DSNUM = 2

BMC96400I MERGE PHASE MESSAGES FOR THE FOLLOWING OBJECT ARE IN FILE AFRPR003
BMC40301I TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I DSNUM = 3

**Figure 131: Example 18--AFRPR001**

BMC40016I MERGE PHASE STARTING 08/22/2014 17:16:49
BMC40301I TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I DSNUM = 1
BMC40350I DATA SET AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P01 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00004 ELAPSED = 00:00:00

BMC40060I PENDING FLAG RECP SET FOR
BMC40345I SPACE = G4GLMQDB.V9LMQTS1
BMC40305I DSNUM = 1
BMC40356I AN IDCAMS 'DELETE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBC.G4GLMQDB.V9LMQTS1.I0001.A001
BMC40356I AN IDCAMS 'DEFINE CLUSTER' OPERATION WAS SUCCESSFUL ON DATASET DEDLCAT.DSNDBC.G4GLMQDB.V9LMQTS1.I0001.A001
BMC40350I DATA SET DEDLCAT.DSNDBD.G4GLMQDB.V9LMQTS1.I0001.A001 WAS SUCCESSFULLY ALLOCATED TO DDNAME SYS00010 ELAPSED = 00:00:00

**Example 18: Using the MAXLSORT default value**

Chapter 5 Examples of NGT Recover jobs  441
Example 18: Using the MAXLSORT default value

Figure 132: Example 18—AFRPR002
**Example 18: Using the MAXLSORT default value**

Chapter 5 Examples of NGT Recover jobs 443
Example 18: Using the MAXLSORT default value

- **00:00:00**
  
  **BMC40905I** SPACE WRITE WAITS = 2066, WAIT TIME = 00:01:01
  
  **BMC40985I** NUMBER OF PAGES READ FROM FULL COPY FOR THIS OBJECT = 428613, USED = 428613
  
  **BMC40790I** NUMBER OF PAGES READ FROM FULL COPY TOTAL = 428613, USED = 428613
  
  **BMC40798I** NUMBER OF PAGES OUTPUT IS = 428613
  
  **BMC40900I** MERGE TABLESPACE STEP COMPLETE
  
  **BMC40593I** MERGE PROCESS COMPLETE, ELAPSED TIME = 00:01:06
  
  **BMC40352I** DDNAME SYS00013 (DSN DEDLCAT.DSNDBD.G4GLMQDB.V9LMQTS1.I0001.A002) WAS SUCCESSFULLY UNALLOCATED
  
  **BMC40352I** DDNAME SYS00011 (DSN AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P02) WAS SUCCESSFULLY UNALLOCATED
  
  **BMC40060I** PENDING FLAG RECP RESET FOR
  
  **BMC40345I** SPACE = G4GLMQDB.V9LMQTS1
  
  **BMC40305I** DSNUM = 2
  
  **BMC40091I** 08/22/2014 17:17:56 SUBPHASE CHECKPOINT TAKEN FOR MERGE PHASE
  
  **BMC40012I** MERGE PHASE COMPLETE. ELAPSED TIME = 00:01:06, TIME SINCE UTILITY START = 00:01:14

- **00:00:00**
  
  **BMC40102I** MERGE TABLESPACE STEP STARTING
  
  **BMC40795I** FULL COPY DATASET IS AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P03
  
  **BMC40799I** MERGE STARTING FOR DATA SET DEDLCAT.DSNDBD.G4GLMQDB.V9LMQTS1.I0001.A003
  
  **BMC40593I** INITIALIZATION COMPLETE, ELAPSED TIME = 00:00:00
  
  **BMC40593I** SPIN COMPLETE, ELAPSED TIME = 00:00:00
  
  **BMC40401I** EXTENT ALLOCATED FOR TABLE SPACE G4GLMQDB.V9LMQTS1
  
  **BMC40305I** DSNUM = 3
  
  **BMC40302I** DSN = DEDLCAT.DSNDBD.G4GLMQDB.V9LMQTS1.I0001.A003
  
  **BMC40593I** SPIN COMPLETE, ELAPSED TIME = 00:00:00
  
  **BMC40400I** START MERGE COMPLETE, ELAPSED TIME = 00:00:00

**Figure 133: Example 18—AFRPR003**

**BMC40181I** RECOVER PLUS FOR DB2 V11.2.00

**BMC40731I** COPYRIGHT BMC SOFTWARE INC. 1991-2015

**BMC91673I** RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001

**BMC40473I** CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM

**BMC40418I** RECOVER PLUS FOR DB2 V11.2.00

**BMC40473I** COPYRIGHT BMC SOFTWARE INC. 1991-2015

**BMC91673I** RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001

**BMC40016I** RECOVER PLUS FOR DB2 V11.2.00

**BMC40473I** COPYRIGHT BMC SOFTWARE INC. 1991-2015

**BMC91673I** RECOVER PLUS TECHNOLOGY IS PROTECTED BY U.S. PATENTS 7,133,884 7,769,718 AND 8,161,001

**BMC40474I** CONTACT BMC SUPPORT AT 1-800-537-1813 OR EMAIL TO SUPPORT@BMC.COM
Figure 134: Example 18—AFROSUM
TABLESPACE RECOVERY

LOG RESOURCE RANGE:
FROM LOGPOINT = X'000000000FB698FB3877' TO LOGPOINT =
X'000000000FB6942B835'

INPUT COPY / SYSCOPY DATA:
2014-08-22 17.14.20.6138 RBA/LRSN X'000000000FB698FB3877' FULL IMAGE

DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P01
SHRLEVEL = REFERENCE SITETYPE = LP DSNUM = 1

UTILITY POINT - WRITE(YES)

OBJECT SUMMARY:
TABLESPACE = G4GLMQDB.V9LMQTS1
PARTITION = 2 OF 3
DBID = X'0539' PSID = X'0002'
ORIGINAL STATUS = RO
STOGROUP = G4GLMQSG PRIQTY = 25000 SECQTY = 50000
ENCODING SCHEME = EBCDIC

INPUT COPY / SYSCOPY DATA:
2014-08-22 17.14.36.2125 RBA/LRSN X'000000000FB698FB3877' FULL IMAGE

DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P02
SHRLEVEL = REFERENCE SITETYPE = LP DSNUM = 2

UTILITY POINT - WRITE(YES)

OBJECT SUMMARY:
TABLESPACE = G4GLMQDB.V9LMQTS1
PARTITION = 3 OF 3
DBID = X'0539' PSID = X'0002'
ORIGINAL STATUS = RO
STOGROUP = G4GLMQSG PRIQTY = 25000 SECQTY = 50000
ENCODING SCHEME = EBCDIC

INPUT COPY / SYSCOPY DATA:
2014-08-22 17.14.46.6627 RBA/LRSN X'000000000FB698FB3877' FULL IMAGE

DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P03
SHRLEVEL = REFERENCE SITETYPE = LP DSNUM = 3

UTILITY POINT - WRITE(YES)

ACCESS(Force)
Example 18: Using the MAXLSORT default value
BMC40721I  PHASE INITIALIZATION:
BMC40721I  TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I  DSNUM = 2
BMC40738I  ALLOCATE INPUT IMAGE COPY FILE
BMC40302I  DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P02
BMC40738I  FILE OPTIMIZATION
BMC40738I  ALLOCATE 2 INPUT FILE BUFFER(S)
BMC40738I  OPEN IMAGE COPY FILE
BMC40738I  ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I  ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I  ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40738I  SET RECOVERY PENDING (RECP) FLAG
BMC40738I  CREATE SPACE
BMC40345I  SPACE = G4GLMQDB.V9LMQTS1
BMC40305I  DSNUM = 2
BMC40738I  ALLOCATE SPACE (READ/WRITE)
BMC40345I  SPACE = G4GLMQDB.V9LMQTS1
BMC40305I  DSNUM = 2
BMC40738I  CONNECT TO SPACE
BMC40738I  ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I  FILE OPTIMIZATION
BMC40738I  ALLOCATE 2 SPACE OUTPUT BUFFER(S)
BMC40738I  ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I  ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40738I  MERGE SPACE
BMC40301I  TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I  DSNUM = 2
BMC40735I  PHASE TERMINATION:
BMC40738I  DISCONNECT FROM SPACE
BMC40738I  DEALLOCATE SPACE
BMC40345I  SPACE = G4GLMQDB.V9LMQTS1
BMC40305I  DSNUM = 2
BMC40738I  FREE PAGE CHECK BUFFER
BMC40738I  FREE SPACEMAP BUFFER
BMC40738I  FREE PASTE BUFFER
BMC40738I  FREE 2 OUTPUT SPACE BUFFER(S)
BMC40738I  FREE PAGE CHECK BUFFER
BMC40738I  FREE PASTE BUFFER
BMC40738I  FREE SPACEMAP BUFFER
BMC40738I  CLOSE IMAGE COPY FILE
BMC40302I  DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P02
BMC40738I  FREE 2 INPUT FILE BUFFER(S)
BMC40738I  DEALLOCATE INPUT IMAGE COPY FILE
BMC40302I  DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P02
BMC40738I  reset recovery pending (RECP) flag
BMC40738I  register recover to current in syscopy
BMC40738I  record restart information

Example 18: Using the MAXLSORT default value

BMC40701I  PHASE:  MERGE
BMC40721I  PHASE INITIALIZATION:
BMC40721I  TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I  DSNUM = 3
BMC40738I  ALLOCATE INPUT IMAGE COPY FILE
BMC40302I  DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P03
BMC40738I  FILE OPTIMIZATION
BMC40738I  ALLOCATE 2 INPUT FILE BUFFER(S)
BMC40738I  OPEN IMAGE COPY FILE
BMC40738I  ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I  ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I  ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40738I  CREATE SPACE
BMC40345I  SPACE = G4GLMQDB.V9LMQTS1
BMC40305I  DSNUM = 3
Example 18: Using the MAXLSORT default value

Chapter 5  Examples of NGT Recover jobs  449

BMC40738I ALLOCATE SPACE (READ/WRITE)
BMC40345I   SPACE = G4GLMQDB.V9LMQTS1
BMC40305I   DSNUM = 3
BMC40738I CONNECT TO SPACE
BMC40738I ALLOCATE PAGE CHECK BUFFER (4096 BYTES)
BMC40738I FILE OPTIMIZATION
BMC40738I ALLOCATE 2 SPACE OUTPUT BUFFER(S)
BMC40738I ALLOCATE SPACEMAP BUFFER (4096 BYTES)
BMC40738I ALLOCATE PASTE BUFFER (4096 BYTES)
BMC40702I  STEP:  MERGE SPACE
BMC40738I MERGE SPACE
BMC40301I   TABLESPACE = G4GLMQDB.V9LMQTS1
BMC40305I   DSNUM = 3
BMC40735I PHASE TERMINATION:
BMC40738I DISCONNECT FROM SPACE
BMC40738I DEALLOCATE SPACE
BMC40345I   SPACE = G4GLMQDB.V9LMQTS1
BMC40305I   DSNUM = 3
BMC40738I FREE PAGE CHECK BUFFER
BMC40738I FREE SPACEMAP BUFFER
BMC40738I FREE PASTE BUFFER
BMC40738I FREE 2 OUTPUT SPACE BUFFER(S)
BMC40738I FREE PAGE CHECK BUFFER
BMC40738I FREE PASTE BUFFER
BMC40738I FREE SPACEMAP BUFFER
BMC40738I CLOSE IMAGE COPY FILE
BMC40302I   DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P03
BMC40738I FREE 2 INPUT FILE BUFFER(S)
BMC40738I DEALLOCATE INPUT IMAGE COPY FILE
BMC40302I   DSN = AFR.MXLTV9.BMCFCPYB.G4GLMQDB.V9LMQTS1.P03
BMC40738I RESET RECOVERY PENDING (RECP) FLAG
BMC40738I REGISTER RECOVER TO CURRENT IN SYSCOPY
BMC40738I RECORD RESTART INFORMATION
NGT Recover concepts

This chapter describes advanced topics including the optimization process that is used by NGT Recover to determine the best approach for recovery and how you can affect performance by specifying particular options or by changing data set allocations.

Recovery of partitioned table spaces and indexes

You can use NGT Recover to recover table space partitions, partitioned indexes, and nonpartitioned indexes in a variety of ways. This section illustrates the following strategies:

- Recovering table space partitions in separate jobs with all jobs running concurrently
- Recovering table space partitions and recovering or rebuilding the partitioned indexes on the same partitions with all jobs running concurrently
- Recovering or rebuilding a nonpartitioned index on multiple partitions in a single job and in multiple jobs
- Rebuilding a nonpartitioned index on multiple partitions in separate jobs with all unload keys jobs running concurrently
- Recovering table space partitions and recovering or rebuilding the related partitioned and nonpartitioned indexes with multiple RECOVER and REBUILD command statements

Recovering multiple table space partitions in separate jobs

NGT Recover provides the maximum possible concurrency for recovering multiple partitions of a partitioned table space in separate jobs.
The following figure shows multiple jobs concurrently recovering the multiple partitions of a partitioned table space. You can submit as many jobs as required for the number of partitions that you have created. However, if archive log tapes are required for the table space recovery, only two jobs may use a particular log range concurrently because there are at most two copies of the archive log. If your archive logs are stored on DASD, this constraint does not apply. Jobs that encounter other users on both tape copies of an archive log must wait for a copy to become available.

Figure 136: Multiple jobs recovering partitions of a table space
Recovering table space partitions and rebuilding a partitioned index in multiple jobs

You can rebuild a partitioned index for a partitioned table space in separate jobs.

The following figure shows multiple jobs recovering the table space partitions and rebuilding the partitions of a partitioned index for a partitioned table. Using separate jobs for the partitions is ideal when there is little or no log to process and the image copies that you have made are on separate data sets.
With sufficient system resources, these jobs may run in less elapsed time than one job takes to recover all of the partitions and rebuild the partitioned index.

Figure 137: Recovering a table space and rebuilding a partitioned index
Rebuilding nonpartitioned indexes with single and multiple jobs

If nonpartitioned indexes are defined on a partitioned table space, you must extract the keys from all partitions of the table space to rebuild the index.

You can do this with a single command statement and job as shown in the following figure.

**Figure 138: Rebuilding a nonpartitioned index with a single job**

You can also use NGT Recover to extract the keys in separate jobs. The index can then be re-created from the key data sets. The following figure shows keys being
extracted from a three-partition table space in three jobs followed by a job to build the index. The sorting is in the UNLOADKEYS jobs.

Figure 139: Rebuilding a nonpartitioned index with multiple jobs

Recovering a partitioned table space and rebuilding a nonpartitioned index

If you need to recover a partitioned table space and a nonpartitioned index, you can combine RECOVER TABLESPACE and RECOVER UNLOADKEYS command statements.

As shown in the following figure, you can set up multiple jobs to do the processing on the individual partitions followed by a RECOVER BUILINDEX job to build the nonpartitioned indexes. If your system has enough resources, these jobs can extract the keys from several partitions concurrently, and may run in less time than running
one job to read all of the partitions. Contention for archive log tapes may cause jobs to wait.

Figure 140: Recovering a partitioned table space and a rebuilding nonpartitioned index

When you run a RECOVER BUILDINDEX job, the JCL must include all of the data sets containing the extracted keys, and use a separate SKEYnn ddname for the file from each UNLOADKEYS job step.

When you run a RECOVER BUILDINDEX job, ensure that no updates or other changes have been made to the table space since the keys were extracted. Otherwise, the integrity of the data cannot be guaranteed. For a discussion, see “Concurrency with RECOVER UNLOADKEYS and RECOVER BUILDINDEX” on page 77.
Recovering a table space and rebuilding a partitioned index and a nonpartitioned index

The LOGSORT/MERGE strategy for recovering a table space allows you to concurrently extract the keys for nonpartitioned indexes and rebuild one or more partitioned indexes.

After running these concurrent jobs, you can build the nonpartitioned index. The following combination of RECOVER TABLESPACE, REBUILD INDEX, and RECOVER UNLOADKEYS command statements illustrates this method:

1 Run jobs 1 and 2 concurrently.

   **JOB 1:**
   ```
   RECOVER TABLESPACE X.Y DSNUM 1
   REBUILD INDEX (X.YCLUS) PART 1
   RECOVER UNLOADKEYS (X.YNCLUS) PART 1
   ```

   **JOB 2:**
   ```
   RECOVER TABLESPACE X.Y DSNUM 2
   REBUILD INDEX (X.YCLUS) PART 2
   RECOVER UNLOADKEYS (X.YNCLUS) PART 2
   ```

2 When jobs 1 and 2 are complete, run job 3.

   **JOB 3:**
   ```
   RECOVER BUILDINDEX (X.YNCLUS)
   ```

Because NGT Recover reads all of the contents of SYSIN before deciding on a recovery plan, the utility can process the RECOVER TABLESPACE, REBUILD INDEX, and RECOVER UNLOADKEYS command statements for each run as a single unit.

**Note**

When you run a RECOVER BUILDINDEX job, ensure that no updates or other changes have been made to the table space since the keys were extracted. Otherwise, the integrity of the data cannot be guaranteed. For a discussion, see “Concurrency with RECOVER UNLOADKEYS and RECOVER BUILDINDEX” on page 77.

For more information, see “Overview of the LOGSORT strategy” on page 458 and “Overview of the UNLOADKEYS and BUILDINDEX strategy” on page 463.

Overview of the LOGSORT strategy

Sorting the log allows NGT Recover to use a LOGSORT/MERGE strategy.
NGT Recover examines the pertinent log ranges and extracts the appropriate log records from the DB2 active and archive logs. NGT Recover then passes these log records to BMCSORT, where they are sorted by object, page number, and log point.

Sorting the log allows the objects to be processed sequentially, which is more efficient than accessing them randomly as in a conventional log apply process. Using the LOGSORT/MERGE strategy can reduce the I/O required by performing the following tasks:

- Combining the MERGE and LOG APPLY phases
- Ensuring that each page is accessed only once

Sorting the log enables you to read multiple log files in parallel, which can reduce the elapsed time necessary to read all of the log records for the run.

**Why is it faster to sort the log?**

Sorting the log records in page number order allows NGT Recover to invoke a merge process that takes a full copy, any number of incremental copies, an input change accumulation file, and the sorted log as input and merges them into a recovered space image.

In data processing terms, this process is analogous to a classic master file update. The full copy is similar to the master file because it was the exact image of the table space at some point in time (this is not strictly true for SHRLEVEL CHANGE copies, which are discussed later). The incremental copies (limited by MAXDRIVES, see “MAXDRIVES integer” on page 115), change accumulation files, and log records are like update files containing transactions to be applied to the master file.

The merge process simply applies the changes to the full copy to produce an updated image. This strategy is much faster than the conventional strategy, which first writes a space to the merged image of the full copy and incremental copies, and applies the unsorted log record updates to the space (a random read-write operation). Many conventional recoveries have long log apply phases.

Because the inputs are in page order for the LOGSORT/MERGE strategy, NGT Recover can create the final version of the page in memory and write the table space once as a sequential write operation. Sequential write processes, when appropriately blocked, buffered, and scheduled, are much faster than random write processes.

**Note**

If your full copies are made inline with the IBM DB2 utilities, NGT Recover must first restore the copy to the space before merging other resources. This action may cause more lengthy recoveries.
When indexes are recovered by using copies and log records, they are handled in the same way as table spaces and can benefit from the LOGSORT/MERGE strategy.

What are the restrictions when using LOGSORT?

Merges that fail cause LOG INPUT to be repeated on restart for log records of the failed merges.

Failure of an image copy that is detected during merge may cause a merge to require additional log records. When additional log records are required, the job must be restarted to complete failed merges.

Does LOGSORT have other advantages or disadvantages?

The LOGSORT strategy can provide other advantages, as follows:

- If keys are being extracted in a table space recovery because of either a REBUILD INDEX or RECOVER UNLOADKEYS request for an index on a table in the table space, NGT Recover can extract these keys during the merge process of a LOGSORT strategy. In some cases, the keys are sent directly to a sort at this point by using the NOWORKDDN option (see “Overview of the NOWORKDDN strategy” on page 463), or the keys are written to the temporary data set specified by the WORKDDN parameter, usually SYSUT1 for REBUILD INDEX, or the SKEYDDN parameter for RECOVER UNLOADKEYS, depending on how you have set your index sort option. By extracting the keys during the merge process, NGT Recover avoids rereading the table space to rebuild the index, as is the case in the conventional strategy. This strategy can save elapsed time.

- Because NGT Recover sees each page in its final form and writes the object sequentially, very little overhead is incurred by simultaneously making image copies. Therefore, with the LOGSORT strategy, NGT Recover can optionally make and register (in SYSIBM.SYSCOPY) up to four copies of a recovered table space.

- The LOGSORT strategy allows the use of input change accumulation files and the simultaneous processing of a RECOVER command statement and an ACCUM command statement for the same object. (For more information about ACCUM, see the R+/CHANGE ACCUM for DB2 User Guide.)

- The LOGSORT strategy allows the use of BACKOUT for point-in-time recovery. This strategy can be considerably faster than when copies are used. (For more information, see “Strategies for point-in-time recovery” on page 466.)

- The LOGSORT strategy allows NGT Recover to read multiple log files concurrently, which may significantly improve elapsed time. (For more information, see “Reading multiple log files concurrently” on page 490.)
The LOGSORT strategy can provide disadvantages, as follows:

- The potential disadvantage of using the LOGSORT strategy is that it may require additional resources if there are a significant number of log records to sort. This situation can be alleviated by the periodic running of jobs to build change accumulation files by using R+/CHANGE ACCUM. For more information about change accumulation files, see the R+/CHANGE ACCUM for DB2 User Guide.

- NGT Recover invokes BMCSORT. The amount and type of resources required for the sort depend on the number of log records to be sorted.

  If you are reading many log files concurrently, the memory usage of the run will increase. Approximately two megabytes of memory (in the extended, private area above the 16-MB line) will be used for each log file that is read concurrently.

  You can control the number of log files read concurrently with the MAXLOGS option. For more information about the additional resources that may be required for the LOGSORT strategy, see “Log sort space requirements” on page 333, “Managing sort performance” on page 474, and “Setting the SMCORE option” on page 477.

---

### Using LOGONLY or LOGAPPLY ONLY with point-in-time recoveries

When a point-in-time recovery is performed, two log points are recorded in SYSIBM.SYSCOPY or BMCXCOPY:

- PIT_RBA: the point to which the table space is recovered
- START_RBA: the point at which the point-in-time recovery was performed

---

**Note**

If you are operating in a data sharing environment, log record sequence number (LRSN) values are stored in SYSIBM.SYSCOPY. LRSNs for PIT_RBA indicate the *ending* point to which the table space is recovered. LRSNs for START_RBA indicate the point at which the point-in-time recovery was performed.
If the log range used for a LOGONLY or LOGAPPLY ONLY recovery includes both the PIT_RBA and the START_RBA from a point-in-time recovery, the log range between those log points is ignored (see the following figure).

Figure 141: How PIT_RBAs and registered log points are used during LOGONLY or LOGAPPLY ONLY recoveries

The FROMLOGPOINT of a LOGONLY recovery is obtained from the log point that is stored in the header page, while LOGAPPLY ONLY obtains it from the syntax.

When the FROMLOGPOINT of a LOGONLY or LOGAPPLY ONLY recovery is greater than the PIT_RBA of a point-in-time recovery and the TORBA of the LOGONLY or LOGAPPLY ONLY recovery is greater than or equal to the START_RBA of the point-in-time recovery (see the following figure), NGT Recover does not allow a recovery to be performed. If NGT Recover attempted this recovery, the restored copy may include some, but not all, updates that are located between the PIT_RBA and the START_RBA. Recovery of the log records found after the START_RBA depends on none of those updates being present. If you must perform such a recovery and you are using a non-DB2 backup, you must select a backup before the PIT_RBA.

Figure 142: When a recovery is not allowed by NGT Recover
Overview of the NOWORKDDN strategy

In a conventional REBUILD INDEX strategy that uses WORKDDN, the keys to build the index are written to a work data set (typically SYSUT1) while they are being extracted from the table space data.

When the extraction is complete, the keys are read back in and sorted, and then the index is built from the sorted key/RID pairs.

However, using the NOWORKDDN option allows the extracted keys to go directly into a sort process. This strategy avoids writing and rereading the key work data set, and no work data set is required to hold the extracted keys. While restartability may be affected (see Table 11 on page 335), the I/O reduction significantly reduces the time that is required to rebuild the index.

If you do not specify WORKDDN, the NOWORKDDN option is the default. NOWORKDDN is required for the rebuild of the partitioning index when used together with the RECOVER UNLOADKEYS strategy for the nonpartitioning indexes.

For more information, see “NOWORKDDN” on page 228.

Overview of the UNLOADKEYS and BUILDINDEX strategy

The rebuilding of large nonpartitioned indexes on partitioned table spaces is a major recovery issue.

To recover these objects with a conventional strategy requires a sequential read pass of all partitions of the table space. In some cases, the amount of time that is required to read the partitioned space and rebuild the index with a conventional strategy is
prohibitive. To facilitate the recovery of these spaces, NGT Recover provides concurrency in the extraction of keys from multiple partitions.

For example, if you have a large 16-partition table space with a nonpartitioned index that must be recovered (perhaps because of media failure), NGT Recover provides an alternative strategy that uses the RECOVER UNLOADKEYS and RECOVER BUILDINDEX command statements. You can submit up to 16 jobs to extract keys from multiple partitions simultaneously. Sixteen jobs are not required. You can submit up to one job per partition. If you have only eight available initiators, you can submit eight jobs, with each job reading two partitions sequentially. Any combination (10-2-4, 4-4-4-4, and so on) of partitions per job is acceptable.

Specify the RECOVER UNLOADKEYS command statement for the nonpartitioned index and specify the PART option to indicate which table space partition is to be used for this key extraction. Include a DD statement in the JCL to specify the work data set to hold the keys. The extracted keys files are sorted and are then input to the RECOVER BUILDINDEX job where the keys are merged to build the index.

The improved concurrency can drastically reduce the elapsed time that is required to rebuild a nonpartitioned index. Because RECOVER UNLOADKEYS performs the sorting, particularly for a large, partitioned space, you may see performance improvements because a number of smaller sorts are done.

You can extract the keys for RECOVER UNLOADKEYS indexes during the table space recovery process, further reducing total elapsed time for combined table space and index recovery. For an example of a combined LOGSORT/MERGE strategy and UNLOADKEYS/BUILDINDEX strategy, see “Example 6: Extracting nonpartitioned index keys” on page 402.
Attempting to recover to a SHRLEVEL CHANGE copy

With NGT Recover, you can specify RECOVER TABLESPACE TOCOPY that uses a SHRLEVEL CHANGE copy.

However, the data on such a copy may not be consistent. This procedure is not recommended and NGT Recover issues a warning to this effect.

For more information, see “TOCOPY specification” on page 202.

Fallback recovery with NGT Recover

Fallback in NGT Recover describes the use of an equivalent or prior image copy when the image copy originally chosen for recovery is not available or is invalid.

In most cases, NGT Recover can fall back to another copy automatically. However, in a few cases where automatic fallback does not occur, it is necessary to restart a failed job.

Note

NGT Recover supports automatic fallback for the following copies:

- Standard image copies and Instant Snapshot copies
- Full and incremental copies
- Copies of table spaces or index spaces

Automatic fallback processing includes the following behavior for NGT Recover:

- During the ANALYZE phase, if NGT Recover detects that the specified image copy is not available, NGT Recover falls back to an equivalent image copy if one is available. If such an image copy is not available, NGT Recover falls back to an earlier image copy. If no prior image copy is available, NGT Recover tries to recover from the log. If the required log is no longer available, fallback fails and the space is unrecoverable.

- If the specified image copy is unavailable when NGT Recover attempts to allocate the copy, the recovery falls back to an equivalent image copy. With automatic fallback, NGT Recover uses an earlier image copy or LOAD REPLACE LOG YES or REORG LOG YES log point to recover the space.

- If the selected image copy is available but is corrupted, NGT Recover does not detect this until after image copy allocation. With automatic fallback, NGT Recover uses an earlier image copy or LOAD REPLACE LOG YES or REORG LOG YES log point to recover the space.
Note
Automatic fallback does not occur if you specify any of the following commands and options:
- INCOPY option on a RECOVER command
- RECOVER UNLOADKEYS command
- REBUILD INDEX command with the WORKDDN option

During automatic fallback, RNGT Recover attempts to recover the specified spaces until all of the spaces are recovered or determined to be unrecoverable. The maximum number of severe errors allowed is defined by the ERRCONT installation option (see “ERRCONT=10” on page 583). The recovery step ends with a completion code of 0 if RNGT Recover successfully recovers, initially or following a successful fallback, all of the spaces specified. If RNGT Recover cannot recover all of the spaces, the completion code is 12.

Message BMC40227 indicates that RNGT Recover is beginning automatic fallback processing. Automatic fallback uses messages BMC40879 to indicate the number of failures before fallback is attempted. For example, the following messages indicate that recovery failed for one space and RNGT Recover is beginning automatic fallback processing to recover the space:

BMC40879I 1 OBJECTS WERE NOT RECOVERED DUE TO SEVERE ERRORS
BMC40227I BEGINNING AUTOMATIC FALLBACK

Strategies for point-in-time recovery
NGT Recover is often used to return a set of table spaces and their indexes to a prior point in time, which you can accomplish by using the TOLOGPOINT/TORBA or TOCOPY options.

NGT Recover allows several innovative ways to speed up and plan for this type of recovery, such as use of the BACKOUT option and use of Instant Snapshot copies.

Using BACKOUT
When you must recover to a point in time, the table spaces and index spaces are normally undamaged and up-to-date, but the spaces contain the results of unwanted updates. For this scenario, NGT Recover provides an additional alternative for point-in-time recovery. The following figure illustrates copies that are taken on a weekend, archive logs that are created during the week, and a batch job that runs on Friday night, updating a set of table spaces and indexes. If this job runs incorrectly, a point-in-time recovery is needed to a point before the job.
A point-in-time recovery normally uses image copies and archive and active logs to recover the table spaces. Unless you make image copies of the indexes, you must rebuild the indexes. BACKOUT allows NGT Recover to use the active logs only in this case. Tape mounts are completely avoided and only the parts of the index and table spaces that need to be updated are read and written.

**Figure 143: BACKOUT recovery uses the spaces and logs**

Preparing for a BACKOUT recovery

All that is required for a BACKOUT recovery is a point of consistency.

You may quiesce your spaces before a batch job and prepare a RECOVER with the BACKOUT option to use in a batch job failure situation. If you want a copy before the batch job to protect against a media failure or for disaster recovery, use NGT Copy with the Snapshot Copy capability. You may start your batch job after the COPY has obtained a quiesce point for your application group. You can send the copies, when completed, to your disaster storage vault. If the batch job fails, you can use the RECOVER with BACKOUT strategy. The batch job can then be restarted, or normal activity can resume without the batch run.

Executing a BACKOUT recovery

The following sample syntax illustrates a RECOVER with BACKOUT:

```sql
RECOVER TABLESPACE PAYROLL.EMPLOYEE
  TABLESPACE PAYROLL.TAXRATES
  TABLESPACE PAYROLL.BENEFITS
  INDEXSPACE PAYROLL.EMPNUMIN
  INDEXSPACE PAYROLL.EMPNAMIN
  INDEXSPACE PAYROLL.TAXSTSIN
  INDEXSPACE PAYROLL.BENEPLIN
  TOLOGPOINT LASTQUIESCE BACKOUT
```

When the BACKOUT option is used for point-in-time recovery, no copies are used. The log records from the current point of the log to the TOLOGPOINT requested are...
used to back out changes. This recovery is often far cheaper in terms of resources than a conventional forward recovery.

When you use the BACKOUT facility for a point-in-time recovery, you can use log records to recover the indexes. Because no copies are needed, you do not need to make copies of indexes to use this facility. For large indexes, this strategy can be much more efficient than rebuilding the indexes from the recovered table space. This facility also allows NGT Recover to read and write only the portions of the table spaces and index spaces that are affected by the log records backed out.

After a BACKOUT recovery of an index, you cannot recover the index to the current point in time. You must perform one of the following actions:

- Execute the BACKOUT with OUTCOPY YES to create a copy to be used in a later recovery from media damage.
- Run BMC Next Generation Technology Copy for DB2 for z/OS after the BACKOUT. You can make a SHRLEVEL CHANGE copy to reduce the impact on your applications.
- Rebuild the index if it becomes damaged before its next image copy.

**When is BACKOUT recovery allowed?**

The following restrictions apply to a recovery that uses the BACKOUT option:

- Your space must be undamaged and up to date to use BACKOUT. You cannot use BACKOUT if a LOAD or REORG occurred at or after the log point that is requested. You can, however, use BACKOUT with a TOLOGPOINT after a LOAD or REORG, effectively recovering to points after such an event even if no image copy was made.

- If a mass delete on a segmented table that is not defined with DATA CAPTURE CHANGES is backed out, NGT Recover will discover that there is data loss if data pages have been reused and will terminate with an error message.

- If a prior point-in-time recovery has occurred, you cannot request a point-in-time recovery between the PIT_RBA and the START_RBA of the earlier recovery.

- Exception statuses such as RECP are not allowed on a space that is subject to BACKOUT. However, if you are restarting the BACKOUT, the RECP status is allowed.

- The BACKOUT option cannot be used with INDEP OUTSPACE, OUTCOPY ONLY, DROPRECOVERY, or OBIDXLAT.

- You cannot use RECOVER INDEX or RECOVER INDEXSPACE through a point in time that is created by BACKOUT.
BACKOUT is not valid for
— LOB spaces
— NOT LOGGED spaces
— Hash spaces

**BACKOUT and indexes**

The BACKOUT feature operates on table spaces and indexes.

When you recover a table space to a previous point in time, you must also recover all indexes that are defined on that table space to the same point in time or you must rebuild them. This recovery or rebuild is required whether or not BACKOUT is used. Because BACKOUT operates on indexes and does not require an image copy, you can usually use it to avoid expensive rebuild operations, even on indexes that are never image copied.

An exception occurs when you recover an individual partition of a partitioned table space to a previous point in time. In this case, you will need to rebuild any nonpartitioned indexes (NPIs) that are defined on the table space.

**BACKOUT recovery and restart**

You can restart a job that was using BACKOUT, but you must not change the TOLOGPOINT. You must not rerun such a job because NGT Recover does not accept the exception statuses, such as RECP, except in restart.

**Using Instant Snapshots**

Intelligent storage devices are available that can provide almost instantaneous copies of data sets at a specified point in time by controlling the placement of data on large arrays of disks.

Restoration of such copies is also nearly instantaneous, which can significantly reduce recovery time. NGT Recover can use such copies, made by the BMC NGT Copy utility with the BMC EXTENDED BUFFER MANAGER (XBM) or SNAPSHOT UPGRADE FEATURE (SUF) products, for the forward recovery of table spaces and indexes.

For more information about Instant Snapshots, see the following BMC documentation:

- EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide for details about Instant Snapshots and supported storage devices
Registration of Instant Snapshots

LP Instant Snapshots are registered as IBM FlashCopy (FC) copies in SYSIBM.SYSCOPY. Other Instant Snapshots are registered in the BMC BMCXCOPY table with an STYPE of V. They are not registered in SYSIBM.SYSCOPY because the DB2 COPY utility does not recognize them. Standard DB2 utilities can only use FlashCopy (FC) copies registered in SYSIBM.SYSCOPY.

If a backup copy (LB or RB) is produced and its associated primary copy (LP or RP) is an Instant Snapshot, the backup is registered in BMCXCOPY, even if it is a standard copy.

SNAP phase for Instant Snapshot recovery

The SNAP phase is used to restore table spaces and indexes by using Instant Snapshots. The SNAP phase interfaces with XBM. The SNAP phase is followed by a MERGE phase if required (if there are logs to apply, if output copies are requested, or if keys are extracted).

For multi-data-set, nonpartitioned spaces where a DSNUM ALL recovery is requested, a single SNAP phase restores all Instant Snapshots, followed by a MERGE phase.

INCOPY FULL SNAPSHOT

The INCOPY FULL specification includes a SNAPSHOT option so that you can use a non-registered Instant Snapshot as an input copy.

Executing an Instant Snapshot recovery

To use Instant Snapshots in recovery, you must specify the XBMID installation option or XBMID on the OPTIONS command.

XBMID names the subsystem from which the requested Instant Snapshots had been made.

The following sample syntax provides an example of the syntax needed for a recovery that uses an Instant Snapshot:

```
OPTIONS XBMID DBXM
RECOVER TABLESPACE PAYROLL.EMPLOYEE
     TABLESPACE PAYROLL.TAXRATES
     TABLESPACE PAYROLL.BENEFITS
     INDEXSPACE PAYROLL.EMPNUMIN
```
Instant Snapshots and migration

When you use Instant Snapshots for migration and specify the OBIDXLAT option or when you specify the IMPORT or MIGRATE command, NGT Recover automatically detects if the OBIDs have not changed and avoids the overhead of reading and writing all of the pages.

Instant Snapshot restrictions

The following restrictions apply to recoveries that use Instant Snapshots:

- Instant Snapshot copies cannot be created with the OUTCOPY YES or OUTCOPY ONLY options.

- When Instant Snapshots are made with NGT Copy for use with NGT Recover for a nonpartitioned, multi-data-set space, GROUP YES is required and all of the copies must be Instant Snapshots if RECOVER DSNUM ALL is specified for the space. For a DSNUM n recovery, these restrictions do not apply.

- If an Instant Snapshot copy is used to recover the object, REDEFINE YES has no effect and the data sets are processed as if REDEFINE NO had been specified.

XBM diagnostics

To help with performance and problem diagnosis associated with Instant Snapshots, if you have EXTENDED BUFFER MANAGER (XBM) version 5.6 or later and you use Instant Snapshots for recovery, you can print the XBM diagnostics in NGT Recover output. To do so, you include an AFRDBG DD in your NGT Recover JCL, for example:

```bash
//AFRDBG DD DUMMY
```

Instant Snapshot use by other BMC utilities

Instant snapshots are recognized and used by other BMC products that access the BMCXCOPY table in which these copies are registered.

- BMC Next Generation Technology Copy for DB2 for z/OS produces these copies for recovery.

- RECOVERY MANAGER for DB2 reports these copies and recognizes them for recovery when generating and optimizing the necessary JCL.
The NGT Copy MODIFY command deletes the Instant Snapshots from BMCXCOPY and the ICF catalog through its standard functions. All commands operate on the Instant Snapshots registered in BMCXCOPY and their associated standard backup copies. However, template-generated copies are standard copies only, not Instant Snapshot copies.

UNLOAD PLUS for DB2 unloads data from these copies.

IBM FlashCopy

NGT Recover can use IBM FlashCopy image copies as a recovery resource.

The FlashCopy image copies are handled in much the same way as BMC Snapshot copies and are processed by the NGT Recover SNAP phase.

A consistent FlashCopy is processed in the same way as a BMC Online Consistent Copy. For more information about Online Consistent Copy, see “Using Online Consistent Copy” on page 61.

**Note**

Note the following considerations:

- NGT Recover supports consistent FlashCopies with log apply when you do not specify INCOPY.

- NGT Recover supports consistent FlashCopies with INCOPY TOCOPY specified.

- NGT Recover does not support consistent FlashCopies when you specify both INCOPY and log apply (RBA or LOGPOINT specified in the INCOPY specification).

An inconsistent FlashCopy is processed as a SHRLEVEL CHANGE copy in the same manner used for inconsistent BMC Snapshot copies.

Sequential copies created from a consistent FlashCopy are processed in the same way as sequential BMC Online Consistent Copies.

Sequential copies created from an inconsistent FlashCopy are processed as SHRLEVEL CHANGE copies.

The OPTION command RESOURCE SELECTION syntax supports FlashCopy (FC) ("RESOURCE SELECTION" on page 121). FlashCopy is also supported in the installation options LOCCPSEL ("LOCCPSEL=(FC,LP,LB)" on page 586) and REMCPSEL ("REMCPSEL=(RP,RB,FC)" on page 591). The default processing order for local copies is FC, LP, LB. The default processing order for remote copies is RP, RB, FC.
The FLASHCOPY option is available to the Non-registered copy or INLOG specification with INCOPY FULL (“Non-registered copy or INLOG specification” on page 188).

The DSNAME description of the INCOPY specification accommodates FlashCopy support (“INCOPY specification” on page 182).

Changing SHAREOPTIONS for striped or extended format data

NGT Recover can read striped active logs or extended format active logs in non-data-sharing DB2 subsystems.

Note
If you do not have striped active logs or extended format logs, no action is required. You only need to change SHAREOPTIONS as described below if you use striped or extended format logs.

To allow NGT Recover to read striped or extended format (usually greater than 2 GB) logs, you must define the logs with SHAREOPTIONS set to (2,3).

Working with ASCII and Unicode data

ASCII data affects the rebuilding of indexes.

The ASCII definition of a table is directly considered in the following tasks:

- Padding of variable length character columns in an index key
- Assignment of a default value to an index created on a new, unpopulated column defined with NOT NULL WITH DEFAULT
- Assignment of a default value to an index created on a new, unpopulated column defined with NOT NULL WITH DEFAULT 'xyz'.

NGT Recover determines the encoding scheme, the proper padding character, and default values to deal with these issues. NGT Recover is fully compatible with ASCII and UNICODE tables.
Managing sort performance

Performance of an NGT Recover recovery is routinely measured by the elapsed time of the total job.

One of the biggest factors that influences elapsed time is sorting log data, index keys, or both. NGT Recover offers many options that enable you to control the efficiency of the sort and customize the sort based on your environment.

Multiple concurrent key sorts provide the greatest performance gain for the product. For each table space, NGT Recover distributes the index keys for all indexes being rebuilt over a user-defined number of sorts and runs these sorts in parallel. For a partitioned table space, if the partitioning index is being rebuilt, NGT Recover does the rebuild of each partition at the completion of the MERGE or UNLOAD for each partition of the table space and this may happen concurrently with the MERGE or UNLOAD for the next partition (if the user-specified number of sorts is not exceeded).

When NGT Recover has input all of the keys to the sort for a group of indexes, NGT Recover then performs the REBUILD operation for each index in a subtask. There could be one subtask for each group of indexes that were sorted in the previous MERGE or UNLOAD phases. So, index REBUILDs can run concurrently.

NGT Recover determines the number of concurrent REBUILDs for each table space during analysis but never exceeds the user-specified number of sorts. Once NGT Recover schedules all of the REBUILDs for a table space in multiple subtasks, it begins the next MERGE or UNLOAD for a different table space if the user-specified number of sorts will not be exceeded.

NGT Recover uses BMC BMCSORT technology for sorts. This technology provides NGT Recover with more control of the sort process than external sort routines provide. This added control helps prevent memory-related problems during the sort process.

NGT Recover allocates the amount of resources to each sort process based on the amount of work that NGT Recover determines that the sort process will perform. NGT Recover also dynamically detects excess available memory and allocates a percentage of them to the sort processes.

The following areas are of major importance to sort performance and can be customized:

- Number of parallel sorts allowed
- Central and expanded storage
- DASD work space
In general, the more central and expanded storage available to the sort, the better the sort performs. In most cases, the NGT Recover and sort installation default values are sufficient, but if the defaults are not appropriate for your needs, you can use NGT Recover options to override them.

**Recommendations for a large number of concurrent sorts**

NGT Recover automatically checks the available memory in the system (total and in use) and estimates the amount of memory needed to do each sort.

If enough memory is available without impacting the system, NGT Recover starts up to MAXKSORT and MAXLSORT sorts concurrently.

If not enough memory is available, NGT Recover automatically reduces the values for MAXKSORT and MAXLSORT to the maximum number of sorts that can run concurrently with the available memory. NGT Recover first reduces MAXLSORT as far as necessary until the value of MAXLSORT is 2. If too many sorts for available memory are still indicated, NGT Recover then reduces MAXKSORT until a manageable number of sorts is reached.

NGT Recover does not reduce MAXLSORT to favor MAXKSORT if no indexes are being rebuilt.

In almost all cases, NGT Recover optimizes the number of sorts automatically.

In the rare case that too many sorts are running and performance is degrading, you can

- Reduce the number of sorts by setting MAXKSORT
- Control the amount of memory used by each sort using SMCORE

**Note**
You can set the SMCORE and MAXKSORT options in the $AFROPTS installation macro or you can code them in the OPTIONS statement in the SYSIN for each NGT Recover job.

**Sort file size estimation**

The sort file size for each sort required is the major factor that affects the distribution of workloads and resources between separate sorts.

In almost all cases NGT Recover can automatically estimate the file size using DB2 statistics, data set size, image copy size, or DB2 log ranges.
If the file size estimates that result from the use of the defaults are inaccurate and cause performance to suffer, NGT Recover offers ways to override these estimates with the NUMREC option.

**Log sort default**

NGT Recover attempts to estimate a file size for a log sort by calculating the number of pages contained in the SYSIBM.SYSLGRNX log range that is required to recover all the objects in a step and assuming there will be one log record per page input to the sort.

Depending on the number of objects in the recovery, the size of log records, and the number of records for other objects, this strategy may or may not be reasonable. The values that are reported by the LOGSCAN command provide a better estimate for NUMREC and AVGRECSZ.

For more information about sorting the log, see “Overview of the LOGSORT strategy” on page 458.

**Key sort default**

NGT Recover can determine index sort work space estimates based on RUNSTATS or real-time statistics.

**Use of RUNSTATS**

For index rebuilds, if the DB2 UT SORT DATA SET ALLOCATION field (UTSORTAL subsystem parameter) is set to NO, NGT Recover uses the value that is stored by RUNSTATS in the CARD column of SYSIBM.SYSTABLES to determine the cardinality of the table. If the index is for a partitioned table space, the number of rows is divided equally among all partitions. If a -1 is in the CARD column, NGT Recover (by default) assumes that RUNSTATS has not been run and calculates an estimated file size. This calculation is based on the size of the table space, the maximum rows per page, and the number of indexes.

*Note*

If you chose the NUMREC CALC method and a -1 is found in the CARD column, and the **underlying table space** for the index is being recovered at the same time, this automatic calculation is not performed. In this case, no file size estimate is made and sort performance could suffer and sort capacity problems could occur or both. For best performance on index sorts, use NOWORKDDN and keep your catalog statistics up to date with BMCSTATS, RUNSTATS, or the statistics option in NGT Copy. (If you use WORKDDN, NGT Recover determines the exact number of records to sort, but I/O to the work file may degrade performance.)
Use of real-time statistics

For index rebuilds with the DSN6SPRM UTSORTAL parameter set to YES, NGT Recover uses real-time statistics by using the TOTALROWS value from table SYSIBM.SYSTABLESPACESTATS for single-table table spaces.

Note
If UTSORTAL is set to YES and you perform a migration, TOTALROWS may not be accurate. If it is not accurate after the migration, you must reset the value to NULL. However, TOTALROWS is generally accurate and you may not need to reset its value in the following situations:
- If you migrate data daily and the size of the space does not change much
- If you include the real-time statistics tables in your migration

Separate index rebuilds

When rebuilding multiple indexes of varying lengths on the same table space and not enough sorts are allowed to distribute the indexes with different length to separate sorts, sort performance is improved if some indexes are rebuilt in separate jobs because NGT Recover pads all keys to the length of the longest key directed to the same sort to avoid a variable length record sort. Rebuilding some indexes in separate jobs also reduces the amount of sort work space that is required.

NGT Recover sort parameters

NGT Recover has several parameters that are set at installation time that affect sorting.

Setting the SMCORE option

NGT Recover provides the SMCORE installation option to give you control, when necessary, over the amount of memory that BMCSORT uses during a recovery job.

SMCORE is specified at installation time and is part of the AFR$OPTS options module. You can override SMCORE at run time by using the OPTIONS command. This option contains two parameters, total memory and below-the-line memory.

Note
BMC recommends that you specify zero for both SMCORE values (as 0K or 0M) to allow the sort routine to determine the optimal amount of storage to allocate above and below the 16-MB line.
The following values tell NGT Recover to determine the appropriate amount of memory to use for each sort based on the following criteria:

- Value that you specified for REGION in either your JCL or system exits
- Amount of memory that is available during optimization
- Number of sorts to process

— Using KSORTSHARE YES, the maximum number of sorts is:
  \[
  \text{MAXKSORT + MAXLSORT}
  \]

— Using KSORTSHARE NO, the maximum number of sorts is:
  \[
  \text{MAXLSORT + (MAXKSORT * MAXLSORT)}
  \]

For more information, see “Setting the MAXLSORT and KSORTSHARE options” on page 481 and “Setting the MAXKSORT option” on page 479.

**Total memory**

The first parameter value of the SMCORE option tells NGT Recover how much total memory, above and below the 16-MB line, that you want BMCSORT to use during a single invocation. You can enter the value as a number kilobytes (using the suffix K), or megabytes (using the suffix M).

BMC recommends that you specify zero (as 0K or 0M) to allow NGT Recover to determine the optimal amount of storage.

In addition to 0 KB or 0 MB, valid values are 4096 KB through 65536 KB (using the suffix K) or 4 MB through 64 MB (using the suffix M). Regardless of whether NGT Recover determines the value for total memory or you specify a value, NGT Recover multiplies this value by the number of required sort processes to get a value for the total memory required for the current job. Depending on the workload and system environment, NGT Recover distributes this total memory among the sort processes for the job.

For example, if you specify 4096 KB and NGT Recover determines that it needs four sort processes for this job, NGT Recover calculates that it needs 16384 KB total memory for the job. If the workload for each sort process is different, NGT Recover invokes BMCSORT for each sort process with varying amounts of memory. Some of these amounts will be lower and some of these amounts will be higher than the 4096 KB that you specified.

The following additional considerations apply to the SMCORE option:

- NGT Recover always attempts to honor the value that you specified. Improper setting could lead to performance degradation and memory shortage. The region size available for your recovery job with the value that you specify for this
subparameter can constrain the number of sort processes that NGT Recover could start. Because the region size must include space for buffers and other required structures, the entire region size is not available for sort processing.

- When you allow NGT Recover to optimize total memory, NGT Recover does not use more than the value of your region parameter and does not use more than some reasonable percent of available memory.

- If you specify an SMCORE value that is larger than your region size or the available memory on the system, NGT Recover will never use more than the value of your region size or the available memory on the system.

**Below-the-line memory**

The second parameter value of the SMCORE option tells NGT Recover how much memory below the 16-MB line that you want BMCSORT to use during a single invocation. You can enter the value as a number kilobytes (using the suffix K), or megabytes (using the suffix M).

BMC recommends that you specify zero (as 0K or 0M) to allow NGT Recover to determine the optimal amount of below-the-line memory to use. In addition to 0 KB or 0 MB, valid values are 256 KB through 4096 KB (using the suffix K) or 1 MB through 4 MB (using the suffix M).

BMCSORT never needs more than 256 KB of memory below the line. Specifying a value greater than this number can limit the number of sort tasks that NGT Recover can start concurrently.

**Setting the MAXKSORT option**

NGT Recover determines the optimal number of sorts executing concurrently, depending on available resources. You can specify the maximum number of concurrent sorts with the MAXKSORT option.

Under normal circumstances, allow NGT Recover to control the number of sorts processed concurrently. If system resources are constrained or other problems arise, you can change the MAXKSORT option to limit the number of sorts that are running concurrently.

The MAXKSORT installation option or the MAXKSORT option in the OPTION command statement determines the maximum number of index key sorts that NGT Recover can run in parallel and the maximum number of indexes (or index partitions) that NGT Recover can rebuild in parallel for each table space. Setting the value of MAXKSORT too low could significantly increase the time required to rebuild indexes. Setting the value of this parameter too high could result in overuse of system resources, which could cause total system performance degradation as well as performance degradation of the recovery job.
For indexes on nonpartitioned spaces, the optimum value for MAXKSORT is the largest number of indexes that will be rebuilt for any table space in the recovery job. For example, if a recovery job rebuilds five indexes for table space A and three indexes for table space B, set MAXKSORT to a minimum value of 5. For this example, if MAXKSORT is set to a minimum of 8, NGT Recover might be able to overlap the index rebuilds for table space A with the recovery of table space B, reducing the elapsed time of the job. If the number of indexes to rebuild is so large that you cannot set MAXKSORT to that number, set MAXKSORT as high as possible. NGT Recover groups indexes with similar key lengths together to form an index group.

NGT Recover displays the index group assigned to each index in message BMC40865I in the Object Summary.

For indexes on partitioned spaces, NGT Recover uses the value of MAXKSORT and attempts to divide partitions into groups to allow parallel unloading of keys. The number of partition groups is balanced with the number of indexes to achieve the largest number of parallel sorts and index rebuilds without increasing the total number of bytes sorted or the size of the largest sort.

As with nonpartitioned indexes on nonpartitioned table spaces, nonpartitioned indexes are placed in index groups. The number of parallel sorts is the number of partitions in a partition group multiplied by the number of index groups. For a partitioning index rebuilt with nonpartitioned indexes, the partitioning index may be assigned to its own index group or may be placed in a group with nonpartitioned indexes. The number of indexes that are built in parallel equals the number of index groups.

For example, assume you have a partitioned table space with 30 partitions and four nonpartitioned indexes with key lengths of 200, 120, 60, and 50. If you specify OPTION MAXKSORT 15, NGT Recover creates three index groups with the 200- and 120-byte keys having their own group and the 60- and 50-byte keys sharing a group. Each partition group contains five partitions, and there are six groups of partitions.

The partitions are assigned to groups based on the estimated number of rows in each partition such that, as much as possible, the partition groups contain an equal number of rows. For this example, there are 15 parallel index key sorts (five partitions in each partition group multiplied by three index groups). Three indexes are built in parallel because there are three index groups.

The maximum value that you can assign to MAXKSORT is determined by the amount of below-the-line and above-the-line memory available in the region in which the NGT Recover job runs. NGT Recover does not allow a value for MAXKSORT that is larger than that which can be supported by the available memory. The formula to determine the maximum value allowed for MAXKSORT is as follows:

\[ a = \frac{\text{above-the-line memory}}{30 \text{ MB}} + 1 \]
\[ b = \frac{(\text{below-the-line memory} - 1.5 \text{ MB})}{300 \text{ KB}} + 1 \]

\[ \text{MAXKSORT} = \min(a, b) \]

For example, in a region with 7.5 MB below the line and 720 MB above the line, the maximum value for MAXKSORT is 21:

\[ a = \frac{720 \text{ MB}}{30 \text{ MB}} + 1 \]

\[ b = \frac{7.5 \text{ MB} - 1.5 \text{ MB}}{300 \text{ KB}} + 1 \]

\[ \text{MAXKSORT} = \min(25, 21) \]

\[ \text{MAXKSORT} = 21 \]

You should also consider the value of MAXLSORT when setting the value of MAXKSORT. For more information, see Setting the MAXLSORT and KSORTSHARE options on page 481.

### Setting the MAXLSORT and KSORTSHARE options

NGT Recover can run multiple log sorts and parallel MERGE phases. SNAP and RESTORE phases (for inline copies) also run in parallel.

The MAXLSORT option, which you can set in the installation options (“MAXLSORT=0” on page 588) or on the OPTIONS command (“MAXLSORT integer” on page 136), specifies how many log sorts can run in parallel and also determines the number of MERGE/SNAP/RESTORE phases that can run in parallel, regardless of log requirements. In the case of parallel phases, the phases run in subtasks and MAXLSORT sets the number of subtasks.

Use of the MAXLSORT option is dependent on the amount of available memory. BMC suggests that a MAXLSORT value of 10 to 12 is practical. The default value of the MAXLSORT installation option is 0, which allows NGT Recover to determine the number of sorts.

If you specify MAXLSORT 1, you set up your job to run as it would in NGT Recover version 8.1.00 and earlier and turn off parallel log sorts and parallel MERGE/SNAP/RESTORE phases. These phases then run serially in the main task in the order in which they are scheduled by the NGT Recover planning component.

NGT Recover must run some phases serially due to constraints such as

- Sharing a DSNUM ALL or cabinet copy
- Using copies on stacked tape
- Sharing a CHANGE ACCUM file
To honor these constraints, the NGT Recover planning component places the phases into groups. Each phase in such groups runs serially in a subtask. NGT Recover distributes the processing of these groups to balance the workload.

When a table space is recovered and associated indexes are rebuilt in the same step, the MERGE phase extracts keys for the indexes. Since each table space generally uses the maximum number of key sorts specified by MAXKSORT, NGT Recover may not be able to run MERGE phases from different table spaces in parallel because each MERGE requires MAXKSORT key sorts. (For more information about MAXKSORT, see “Setting the MAXKSORT option” on page 479.) In this case, you use the KSORTSHARE option, which you can set in the installation options (“KSORTSHARE=YES” on page 586) or on the OPTIONS command (“KSORTSHARE” on page 137), to specify if key sorts are shared among NGT Recover table space recoveries (MERGE phases) running in parallel.

When you specify KSORTSHARE YES, which is the default value, up to MAXKSORT key sorts are shared at execution. Key sorts can only be used by one table space at a time. With KSORTSHARE YES, NGT Recover generates UNLOAD phases to extract keys. However, during execution, if a sufficient number of key sorts is available when the first MERGE phase for a table space is started, the MERGE phase extracts the keys and the UNLOAD phase is skipped.

When you specify KSORTSHARE NO, each MERGE phase has its own set of key sorts and NGT Recover can have up to MAXKSORT * MAXLSORT key sorts active at any given time. Since the number of sorts that can be active in a system is fairly small - usually no more than 30 - a value of NO for this option may severely limit the number of recovery operations that NGT Recover can perform in parallel when index rebuilds are also requested.

If the values that you specify for MAXLSORT and MAXKSORT result in too many sorts, NGT Recover reduces the value of MAXLSORT until the number of sorts can be accommodated. This reduction in MAXKSORT and MAXLSORT is only necessary when you specify KSORTSHARE NO.

**Setting the RESINV parameter**

The NGT Recover RESINV parameter is used to specify the amount of virtual storage to be reserved below the 16-MB line for program-invoked sorts. This parameter is specified at installation time. NGT Recover recommends that this parameter always be specified as RESINV 0K, which is the default. This value allows the sort utility’s installation default to be used.

**Using the SORTNUM and SORTDEVT parameters**

The SORTNUM parameter controls the actual number of sort work files that are allocated. The SORTDEVT parameter controls the device type to which the sort work files are allocated. These parameters control where each of the sort work files are
allocated when dynamically allocating files. For a summary of information about SORTNUM and SORTDEVT, see the following table.

Table 24: Parameters for DASD Work Space

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>OPTIONS parameter?</th>
<th>Installation option?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORTDEVT</td>
<td>SYSDA</td>
<td>Dynamic sort work unit name</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SORTNUM</td>
<td>0</td>
<td>Number of sort work files</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Other sort parameters

Further control of the sort is available through the NGT Recover OPTIONS command.

See “SORT specification description” on page 113. The OPTIONS command includes the following options:

- SORTDIAG, which turns diagnostic messages on and off
- SORTDYN, which controls and overrides installation defaults for sort work

Note

BMC strongly recommends that you not change the default value for SORTDYN.

Allocation of sort work space

You can allocate sort work space by using dynamic allocation, hard-coded JCL sort work files, or both.

It is critical that you have the right amount of space allocated to handle all of the data without causing performance degradation or abends.

Common problems occur because of inefficient placement of the sort work files or not allocating enough space because bad file size estimate is passed to the sort. In most cases, letting BMCSORT dynamically allocate sort work spaces provides good performance.

Note

To determine the size of sort work data sets, see “Index sort work space requirements” on page 330.
When you hard-code sort work files in JCL, use the following guidelines:

- Allocate in cylinders.
- Allocate all sort work files on one device type, preferably the device with the fastest data transfer rate.
- If possible, allocate sort work files on different volsers across different channel paths.
- Do not use VIO data sets for sort work.
- Allocate enough sort work space in primary storage to contain all input data.
- When sorting multi-gigabyte files, prefer direct JCL allocation of sort work files to dynamic allocation because of the large number of resources needed. Depending on the file size, allocate between four to twelve sort work data sets on separate devices and, preferably, an equivalent number of channel paths for optimal performance.

While BMCSORT allows the allocation of up to 32 hard-coded sort work files for any one sort, hard-coding more than 12 sort work files can cause performance to degrade because of high overhead. Start by allocating four sort work files, and allocate more only if you must because of problems in obtaining extents.

**Strategies for using copies**

This topic describes strategies for using image copies and cabinet copies.

**Image copy data set contents**

An image copy data set that is not a cabinet copy never contains pages from more than one table space or index space. However, using DSNUM ALL in the image copy utility (the default) makes a copy for all of the partitions or data sets of a table space. You may override this option by making copies by using DSNUM integer or by using the BMC NGT Copy product DSNUM PART feature and dynamic allocation.

**Cabinet copy data set contents**

A cabinet copy contains copies from multiple table or index spaces. A cabinet copy may contain DSNUM ALL copies of copies by data set.
Using copies containing multiple data sets

When you need to recover a partitioned or multi-data-set space, use of a copy containing all of the data sets has the following ramifications:

- An attempt to recover the data sets in separate jobs will cause contention for this copy.

- For image copies, the recover utility must read the pages of all previous data sets to recover a specific data set. For cabinet copies, NGT Recover positions to the desired data set.

- If you always recover all of the data sets or have a very small partitioned space and have no need to run separate concurrent recoveries for data sets, an image copy for all data sets may be appropriate. If you have several small table or index spaces that need to be recovered together, a cabinet copy may be appropriate.

- The use of index-controlled partitioning or table-space-controlled partitioning has no ramifications for multi-data-set processing.

For more information about multi-data-set processing for a specific NGT Recover syntax option, see the option description in “NGT Recover syntax” on page 83.

Using data sets stacked on tape

NGT Recover accommodates stacked tape input copies by appropriately ordering the phases of recover and using the data sets in order without dismounting wherever possible.

DD statements for the input image copy data sets should not be included in the JCL; NGT Recover will dynamically allocate them in the correct order. In contrast, DB2 RECOVER requires DD statements with appropriate volume references and retention and label ordering of table spaces in the statements.

You can create stacked copies with the DB2 COPY utility by ordering copy statements and coding appropriate references in the JCL or by using the dynamic allocation capabilities of BMC Next Generation Technology Copy for DB2 for z/OS. If you use wildcarding with NGT Copy, stacked data sets for table spaces are ordered by database name, table space name, and data set number or partition.

NGT Recover can also create output stacked image copies. If DD statements are coded in the JCL for the output image copies, they determine the output stacking order. If dynamic allocation is used, NGT Recover orders the output stacked copies to match the order of any input stacked copies. If no input stacked copies are used, NGT Recover follows the NGT Copy ordering conventions for the output stacked copies.
To recover a space, NGT Recover requires parallel access to all required copies for a merge operation. NGT Recover does not support recovery requests where two or more required copies, such as a full and an incremental copy, are on the same stacked tape. NGT Recover issues the following message in this case:

BMC40087S REQUIRED INPUT COPIES SHARE THE STACKED TAPE AND CAN NOT BE USED FOR MERGE OPERATION

Ensure that input copies that are required to recover any single space do not share a stacked tape.

If you are recovering table spaces and indexes that were copied with NGT Copy by using wildcarding and specifying copies for all partitions, the following example illustrates the order of the stacking supported:

```
DB.TS DSNUM ALL
DB.IXP PART 1
  
  
DB.IXP PART n
DB.IXN
```

In the example, the following representations are made:

- DB.TS is a partitioned table space.
- DB.IXP is a partitioning index space.
- DB.IXN is a secondary index space.

If you use stacked image copy tapes and also plan to use R+/CHANGE ACCUM to create change accumulation files, you can use BMC products to coordinate the order for maximum efficiency. R+/CHANGE ACCUM orders log records for multiple table spaces by database name, table space name, and DSNUM. Log records for table spaces and indexes are ordered like the preceding copies made by NGT Copy.

### Using image copies and log records for indexes

This topic describes some of the strategies that use image copies and log records to recover indexes.

**INDEXLOG AUTO and IXSIZE**

NGT Recover, NGT Copy, and RECOVERY MANAGER have options that automate the creation of index image copies and the use of index image copies in recovery.
With IXSIZE option in NGT Copy you can specify a threshold at which index copies will be made. IXSIZE is also supported by RECOVERY MANAGER.

By using the INDEXLOG AUTO option that is supported by NGT Recover and RECOVERY MANAGER, you can specify the use of an index copy if one is available for recovery. If recovery from an image copy is not possible, NGT Recover and RECOVERY MANAGER will fallback to rebuild the index.

Making and using copies with NGT Copy

NGT Copy and NGT Recover coordinate copies of indexes by registering events in the BMCXCOPY table. Registration of these events in BMCXCOPY is supported for all supported versions of DB2.

For more information, see the examples in “Examples of NGT Recover jobs” on page 367. You can, for example, recover an entire application by using TOCOPY LASTCOPY if NGT Copy has copied all of the objects.

Note
Copies of indexes with the NGT Copy attribute are registered in the SYSIBM.SYSCOPY table. NGT Recover can also use these copies for recovery.

Using log records for indexes

NGT Recover can use index log records for recovery.

For media failure recovery, recovery after a failed LOAD or REORG, or disaster recovery situations, it is necessary to make copies of the index spaces by using NGT Copy, the DB2 COPY utility, DSN1COPY, or pack dumps. If you make copies with NGT Copy or the DB2 COPY utility, they can be used automatically with NGT Recover and log records rolled forward from the registration point. DSN1COPY and pack dump copies must be restored to the space and NGT Recover LOGONLY used to apply log records.

For point-in-time recoveries where the spaces are intact and no LOAD or REORG occurred after the point in time, you can use the NGT Recover BACKOUT feature to reverse index log records to the point in time.

Using R+/CHANGE ACCUM with NGT Recover

The following information pertains to the BMC R+/CHANGE ACCUM for DB2 product.
This product is designed to work with NGT Recover to streamline normal and disaster recovery processes. R+/CHANGE ACCUM accumulates log record data for a specified object or group of objects. Log record data for specified objects is extracted and stored in a change accumulation file. For detailed information about using this product, see the R+/CHANGE ACCUM for DB2 User Guide.

Using change accumulation files for multiple objects

Using R+/CHANGE ACCUM, you can select several specific table spaces and define them as a group.

When you run a R+/CHANGE ACCUM batch job on the group, you create a change accumulation file of log record data for all of the table spaces in the group. You may elect to include log record data for all of the indexes on all of the table spaces in the group.

You may want to create change accumulation files for multiple objects if you are writing change accumulation files to tape or want to avoid creating too many disk data sets. Objects defined as a group are ordered alphabetically by database name and then table space name. Data sets or partitions are ordered within the same space.

When you need to recover an object that shares a change accumulation data set with other objects, consider the following ramifications:

■ An attempt to recover the objects on a file in separate jobs will cause contention for this change accumulation file.

■ NGT Recover must read the log records of all previous objects (sequentially) to recover a specific object.

If you always recover all of the objects or do not anticipate many log records between copies and have no need for separate recovers for the objects, a change accumulation file for multiple objects may be entirely appropriate.

You should attempt to coordinate stacked tape copies and change accumulation containing multiple objects so that, if you use them, you combine objects in a similar order. Target table spaces (table spaces targeted for change accumulation) are ordered by the following items:

■ Database name
■ Table space name
■ Data set number

Indexes for each table space follow the table space in order by partitioned indexes first, and then nonpartitioned indexes, in order by database name and index space name. You should use this order when stacking image copies on tape.
The BMC BMC Next Generation Technology Copy for DB2 for z/OS product orders objects in a similar fashion when using wildcards and dynamic stacking. NGT Recover optimizes access for shared resources, but cannot create an optimal plan if both stacked copies and change accumulation files for multiple objects are used and the stacking is not alphabetical.

Using separate change accumulation files for separate data sets

If you generate separate change accumulation files for separate data sets of a table space, you should use caution in also combining other objects in the same change accumulation file.

Some situations could cause NGT Recover to create a less than optimal plan for recovering objects.

If a particular data set of a table space shares a change accumulation file with other objects and NGT Recover attempts to use a copy including all data sets or to unload keys for a nonpartitioned index on a partitioned space, the plan may use more tape drives or have to reread parts of a copy or change accumulation file to process.

Planning recovery resources

Build a plan for grouping copies on tape, grouping objects on change accumulation files, and using copies for all data sets of an object that allows NGT Recover to use all resources effectively.

NGT Recover will attempt to

- Use shared resources in order (copies including all data sets in data set number order; input change accumulation files in order by database name, table space name, and data set number; input stacked copies by file sequence number)
- Group use of shared resource without having intervening steps that use other resources

NGT Recover must process all data sets of a partitioned table space together when a nonpartitioned index is being rebuilt or when multiple partitions are unloaded.

When change accumulation files are being created, the batch process must process the objects in the grouping order: database name, table space name, and data set number. Similarly, NGT Recover cannot alter a stacking order for output copies that is given by your JCL.
Tips on grouping recovery resources

This section provides tips on grouping recovery resources.

Combining copies for multiple objects with change accumulation for groups

If you are making a copy of all data sets of a space and also making change accumulation files for the same space, make the change accumulation file for all data sets in the space.

If you are stacking copies and building change accumulation files for groups of objects, you should coordinate the objects in the change accumulation group with the stacked copies. Include all of the stacked objects or a subset in the change accumulation group. Stack the copies in alphabetical order by database name and table space name with indexes included as indicated in “Using change accumulation files for multiple objects” on page 488.

Handling large partitioned objects with separate change accumulation files and copies

If you have very large table spaces that are partitioned, make specific data set (partition) copies and do not stack them with other partitions or other table spaces.

Create a change accumulation group for each partition if the change accumulation file will be tape or if a large numbers of updates will occur between image copies.

If the change accumulation file will be disk, and there are not a large number of updates between copies, it may be appropriate to have all of the partitions grouped into the same change accumulation group. This plan allows a straightforward recovery of any partition or concurrent jobs on the partitions.

Reading multiple log files concurrently

NGT Recover can reduce recovery elapsed time by using multiple log readers with the MAXLOGS option.

Rules for using the MAXLOGS option to indicate the number of log readers are as follows:

- Memory usage
  Memory usage will increase by approximately 1.8 MB for each concurrent log reader execution to a maximum of 20MB. For example, if a job has specified
MAXLOGS 4, an additional 7.2 MB of storage will be required during the LOG INPUT phase in an NGT Recover job.

- Setting MAXLOGS and MAXDRIVES

Use the following algorithm to calculate a first try at values for MAXLOGS and MAXDRIVES if you have archive logs on tape (not including logs migrated to tape with DFHSM):

1. Divide the number of tape drives that you have available for recoveries by the number of concurrent NGT Recover jobs that you plan to run. This number is your value for MAXDRIVES. Do not reduce MAXDRIVES below the maximum number of image copy and change accumulation tapes (input and output) for one object.

2. Add the higher of 4 or the number of DASD log files that you anticipate reading to MAXDRIVES to calculate your value for MAXLOGS.

If all your log files are on DASD or are migrated by using DFHSM or a similar product, use values of 0 (zero) for MAXDRIVES and 6 for MAXLOGS. MAXDRIVES 0 leaves the number of tape drives unconstrained.

Allocating output image copy data sets dynamically

NGT Recover uses the OUTPUT command for dynamic allocation of output image copy data sets. The dynamic allocation feature allows you to make image copies of spaces without including DD statements in the JCL. Instead of using DD statements (each of which provides a physical description of only a single data set) you can use the directives (output descriptors) to provide a logical view of how copy data sets are to be created.

The effects of allocating copy data sets dynamically with NGT Recover follow:

- Eliminate large, complex DD statements
- Greatly simplify tape stacking
- Restart a failed job automatically (no JCL or other statements to change)
- Release unused space when a copy data set is closed
- Use symbolic variables and generation data groups (GDGs) to assist in data set name generation

Note

Dynamic allocation of output copies is required for spaces with more than 254 partitions.
Because an output descriptor is not directly associated with a particular data set, you can use an output descriptor to describe multiple copy data sets. (For JCL examples, see “Examples of NGT Recover jobs” on page 367.)

Using copy data set output descriptors

An output descriptor describes the general characteristics of the copy data set, whether it is a disk data set or a tape data set.

These characteristics include:

- Disk or tape unit name
- Operating system cataloging requirements for the data set
- Model data control block (DCB)
- Generic data set name
- Largest number of volumes expected to be used
- SMS class information
- For disk data sets:
  - Disk space information
  - Lists of volumes
- For tape data sets:
  - Stacked tape indicator
  - Optional data compression
  - Data set retention period
  - Data set expiration date

An output descriptor specifies a disk data set or a tape data set. It cannot specify both.

When you want to use dynamic allocation, use an OUTPUT command statement in your SYSIN data set to specify allocation parameters. The OUTPUT command statement must precede your RECOVER command statement as in the following example:

```plaintext
OPTIONS INDEXLOG YES ANALYZE YES OUTCOPY ASCODED
OUTPUT SYSCOP1 UNIT CART STACK YES DSNAME RDAMSM.IC&TYPE.&DB.&TS.P&DNUM(+1) OUTPUT SYSCOP2 UNIT CART STACK YES DSNAME AFR.IC&TYPE.&DB.&TS.P&DNUM(+1) RECOVER TABLESPACE AFRDB01.TS01 OUTCOPY ONLY OUTCOPYDDN(SYSCOP1,SYSCOP2) INDEXSPACE AFRDB01.IX01 OUTCOPY ONLY OUTCOPYDDN(SYSCOP1,SYSCOP2) INDEXSPACE AFRDB01.IX02 DSNUM 1 OUTCOPY ONLY
```
“GDGs and symbolic variables in data set name construction” on page 493 provides more information.

The preceding example allocates all image copies based on the output descriptors for SYSCOP1 and SYSCOP2. Data set names are as follows:

- RDAMSM.ICLP.AFRDB01.TS01.P00(+1)
- AFR.ICLB.AFRDB01.TS01.P00(+1)
- RDAMSM.ICLP.AFRDB01.IX01.P00(+1)
- AFR.ICLB.AFRDB01.IX01.P00(+1)
- RDAMSM.ICLP.AFRDB01.IX02.P01(+1)

If the copies are being made by partition, you do not need to specify a DD name prefix, as you do with JCL-allocated copies. All partitions of the output copy can be allocated with the same OUTPUT descriptor.

For more information, see the syntax description in “OUTPUT syntax” on page 143.

### GDGs and symbolic variables in data set name construction

You can use generation data set groups (GDGs) and symbolic variables to simplify the task of data set name construction when you use dynamic allocation with NGT Recover.

You can use a GDG and symbolic variables together in a data set name.

### Using GDGs

The GDG format that you use in data set name construction is the same as the format that you use in JCL when you use DD statements to allocate your copy data sets.

When dynamic allocation is used, NGT Recover also provides the option of specifying an input data set, AFRGDG, to provide control cards to be used to define the GDG base if it does not already exist. This data set must contain the control cards to perform an IDCAMS DEFINE, and the symbolic variable, &BASE, which NGT Recover replaces with the GDG base name. For example:

```
DEFINE GDG(NAME(&BASE) LIMIT(7) - NOEMPTY NOSCRATCH)
```
Using symbolic variables

You can also use symbolic variables when you specify a data set name in an output descriptor.

You can represent the variable elements shown in Table 25 on page 495 by using symbolic variables.

You can specify any or all nodes of a data set name by using symbolic variables as in the following example:

```
DSNAME &UID.&TS.&TYPE
```

This example generates data set names containing the ID of the user making the copies, the space being copied, and the type of copy. Another example follows:

```
DSNAME NEWYEAR.&DB.&TS
```

This example combines a real node name with symbolic variables to generate a data set name.

Symbols for numeric variables (&DATE, &TIME, &JDATE, &YEAR, &MONTH, &DAY, &JDAY, &HOUR, &MINUTE, &SECOND, &SEQ, &DSNUM, &LDSNUM, &PART, and &LPART) must be prefixed by a Latin alphabetic character. In the following example, the first statement causes errors, while the second is correct:

- Incorrect: DSNAME &DB.&TS.&DATE
- Correct: DSNAME &DB.&TS.RP&DATE

Although you can prefix a symbolic variable with an alphabetic character, you cannot append characters. For example, XX&TS is valid, but &TSXX is invalid. &TS.XX is valid.

You can also use substrings of symbolic variables in data set names in the format `symbolicVariable(n,m)`, where the substring starts with character `n` for a length of `m` characters. In the following example, 2 is the substring starting position and 4 is the substring length.

```
&DBNAME(2,4)
```
You can use symbolic variables and substrings in:
- OUTCOPYDSN
- RECOVERYDSN
- INCOPY MODEL `dataSetName`
- INCOPY MODEL `dataSetName` followed by TOCOPY LASTCOPY

You cannot use substrings in:
- TOCOPY `dataSetName`
- INDEP OUTSPACE MODEL `dataSetName`

You can use symbolic variables with GDGs simply by appending the generation number in parentheses in the usual way. For example, &TS(+1).

Table 25: Symbolic variables for specifying data set names

<table>
<thead>
<tr>
<th>Symbolic variable</th>
<th>Description</th>
<th>Length of result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;JOBNAME</td>
<td>JOB name used in the JCL</td>
<td>8 bytes maximum</td>
</tr>
<tr>
<td>&amp;STEPNAM</td>
<td>STEP name used in the JCL (PROC names are ignored)</td>
<td>8 bytes maximum</td>
</tr>
<tr>
<td>&amp;VCAT</td>
<td>VCAT name for the space</td>
<td>8 bytes maximum</td>
</tr>
<tr>
<td>&amp;DB</td>
<td>Database containing the space</td>
<td>8 bytes maximum</td>
</tr>
<tr>
<td>&amp;TS</td>
<td>Table space or index space</td>
<td>8 bytes maximum</td>
</tr>
<tr>
<td>&amp;SP</td>
<td>Table space or index space</td>
<td>8 bytes maximum</td>
</tr>
</tbody>
</table>
| &DSNUM, &PART     | Data set or partition being copied               | 2 bytes for 0 - 99  
                          |                   | 3 bytes for 100 - 999  
                          |                   | 4 bytes for 1000 - 4096  
| &LDSNUM, &LPART   | Data set or partition being copied               | 3 bytes for 000-999  
                          |                   | 4 bytes for 1000-4096   
<p>| &amp;USERID or &amp;UID   | Job or TSO user ID                               | 7 bytes maximum    |
| &amp;SSID             | DB2 subsystem ID                                 | 4 bytes maximum    |
| &amp;ATTACH           | DB2 group attachment name or subsystem ID        | 4 bytes maximum    |
| &amp;DATE             | Current date (in the form YYMMDD)                | 6 bytes maximum    |
| &amp;TIME             | Current time (in the form HHMMSS)                | 6 bytes maximum    |
| &amp;JDATE            | Current Julian date (in the form YYDDDD)         | 5 bytes maximum    |
| &amp;YEAR             | Current year (in the form YY)                    | 2 bytes maximum    |
| &amp;MONTH            | Current month (in the form MM)                   | 2 bytes maximum    |</p>
<table>
<thead>
<tr>
<th>Symbolic variable</th>
<th>Description</th>
<th>Length of result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DAY d,g</td>
<td>Current day (in the form DD)</td>
<td>2 bytes</td>
</tr>
<tr>
<td>&amp;JDAY d,g</td>
<td>Current Julian day (in the form DDD)</td>
<td>3 bytes</td>
</tr>
<tr>
<td>&amp;HOUR d,g</td>
<td>Current hour (in the form HH)</td>
<td>2 bytes</td>
</tr>
<tr>
<td>&amp;MINUTE d,g</td>
<td>Current minute (in the form MM)</td>
<td>2 bytes</td>
</tr>
<tr>
<td>&amp;SECOND d,g</td>
<td>Current second (in the form SS)</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>
| &ICTYPE          | Type of image copy:  
  ■ F for FULL YES  
  ■ I for FULL NO  
  ■ A for FULL AUTO or CHANGELIMIT  
  ■ D for FULL DSN1COPY | 1 byte |
| &UTIL            | Utility ID | 8 bytes maximum |
| &SEQ             | A sequential number that restarts at 1 at the beginning of each job step execution | 2 bytes |
| &TYPE            | Type of output being produced:  
  ■ LP for local site primary  
  ■ LB for local site backup  
  ■ RP for recovery site primary  
  ■ RB for recovery site backup | 2 bytes |
| &PART4           | Partition for data set allocation  
  You can use this variable for any data set. NGT Recover generates 4-character partition numbers as follows:  
  Partition 1 = 0001  
  Partition 10 = 0010  
  Partition 100 = 0100  
  Nonpartitioned = 0000  
  Example:  
  ABC.DSN1.DA.&DB.&TSIX..P&PART4  
  In this example, NGT Recover generates the following 4-character partition number for partition 496:  
  ABC.DSN1.DA.DBNAME.TSNAME.P0496 | 4 bytes for table spaces with 4096 partitions or fewer |
<table>
<thead>
<tr>
<th>Symbolic variable</th>
<th>Description</th>
<th>Length of result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;PART5</td>
<td>Partition for data set allocation&lt;br&gt;You can use this variable for any data set.&lt;br&gt;NGT Recover generates 5-character partition numbers as follows:&lt;br&gt;Partition 1 = 00001&lt;br&gt;Partition 10 = 00010&lt;br&gt;Partition 100 = 00100&lt;br&gt;Partition 1000 = 01000&lt;br&gt;Nonpartitioned = 00000&lt;br&gt;Example:&lt;br&gt;ABC.DSN1.DA.&amp;DB.&amp;TSIX..P&amp;PART5&lt;br&gt;NGT Recover generates the following 5-character partition number for partition 4096:&lt;br&gt;ABC.DSN1.DA.DBNAME.TSNAME.P04096</td>
<td>5 bytes for table spaces with 4096 partitions or less</td>
</tr>
<tr>
<td>&amp;INST</td>
<td>Instance number, with valid values of 1 or 2</td>
<td>1 bytes</td>
</tr>
<tr>
<td>&amp;UNIQ or &amp;UQ</td>
<td>1- to 8-character value, based on the system clock, that is used to generate unique copy data set names&lt;br&gt;The first character is always an uppercase letter. Each remaining character is either an uppercase letter or a numeral from 0 through 9.</td>
<td>8 bytes maximum</td>
</tr>
<tr>
<td>Symbolic variable</td>
<td>Description</td>
<td>Length of result&lt;sup&gt;a&lt;/sup&gt;&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>a</td>
<td>Any trailing blanks in the result are removed.</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>The maximum total length allowed for a data set name is 44 bytes.</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>&amp;TS for an index copy is the index space name. Using &amp;TS is supported so that a single data set name can be specified for a group containing both table spaces and indexes.</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>Symbols with a numeric result must be prefixed by one or more alpha characters.</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>When used in an INDEP OUTSPACE MODEL, &amp;DSNUM and &amp;PART resolve to a 4-byte partition identifier for the data set node using the following scheme:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ A001 through A999 for partitions 1 through 999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ B000 through B999 for partitions 1000 through 1999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ C000 through C999 for partitions 2000 through 2999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ D000 through D999 for partitions 3000 through 3999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ E000 through E096 for partitions 4000 through 4096</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>This is the group attachment name if one is used as an NGT Recover parameter; otherwise, the subsystem ID is used.</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>The values for these variables are assigned when the output copy data set is allocated.</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>Longer utility IDs are truncated to 8 characters.</td>
<td></td>
</tr>
</tbody>
</table>

### Stacking copies on tape

When you dynamically allocate copy data sets, you can use the STACK YES option to have all copy data sets allocated with the same tape OUTPUT descriptor stacked on tape.

The stacking order is determined by the order of the input image copies, if any. If no input stacked copies exist, NGT Recover uses the default stacking order that is used by NGT Copy wildcard processing. If there is a mix, the copies with corresponding stacked input image copies are stacked first, followed by the remaining copies in default order.

When you dynamically allocate copy data sets, you can optionally stack output copies of the same type (LP, LB, RP, or RB) contiguously on the same set of tapes.

---

**WARNING**

If you are using TMM (Tape Mount Management), be aware that TMM intercepts any data set allocation whether dynamic or otherwise. If you want the copies on tape and use STACK YES with TMM, add the NGT Recover program AFRMAIN to the TMM exclusion list.
When you want to stack copies of more than one type on tape (such as local site primary and local site backup), you must use a different output descriptor for each type and stack the copy types on different tape units. For example, the following command statements stack local site primary copies of table spaces A.B, C.D, and E.F contiguously on tape unit CARTLP and the corresponding local site backup copies contiguously on tape unit CARTLB:

```
OUTPUT LPCOPY UNIT CARTLP ... options ... STACK YES ...
OUTPUT LBCOPY UNIT CARTLB ... options ... STACK YES ...
RECOVER TABLESPACE A.B ..... options
   OUTCOPYDDN(LPCOPY,LBCOPY) ..... more options
RECOVER TABLESPACE C.D ..... options
   OUTCOPYDDN(LPCOPY,LBCOPY) ..... more options
RECOVER TABLESPACE E.F ..... options
   OUTCOPYDDN(LPCOPY,LBCOPY) ..... more options
```

### Automatic sizing of output copies dynamically allocated on DASD

You can use NGT Recover to automatically size DASD output copies dynamically when the primary allocation quantity is not specified by the SPACE option.

Automatic sizing occurs if all of the following conditions exists:

- You are using OUTCOPY (OUTCOPY specification on page 193).
- The copy is dynamically allocated (using the OUTPUT command).
- You did not specify the primary quantity with the SPACE option (“SPACE (primary, secondary) allocation unit” on page 148).
- You specify either the AUTOSIZE=YES installation option (“AUTOSIZE=YES” on page 578) or YES for the AUTOSIZE option on the OPTIONS command (“AUTOSIZE” on page 138).

Optionally, you can use the MAXPRIM option on the OUTPUT command to set a maximum amount of disk space (in the units specified by SPACE) to allocate as primary space. Valid values are 0 through 65535. A nonzero value for MAXPRIM establishes an upper limit for primary space allocation, while a value of zero specifies no limit.

NGT Recover calculates a value for the primary allocation and uses it if it is less than MAXPRIM. Then NGT Recover calculates the secondary allocation as 20 percent of the primary allocation.
If the calculated value for the primary allocation exceeds MAXPRIM, NGT Recover uses the MAXPRIM value. The secondary allocation is based on the remainder of (calculated primary allocation - MAXPRIM).

When automatic sizing occurs for OUTCOPY, the following message is in the MERGE phase output indicating the allocation values that NGT Recover calculated:

```
BMC960091 ALLOCATING OUTPUT IMAGE COPY FILE WITH CYL(15,3)
```

## Using NGT Recover with data sharing

Before you convert from a non-data-sharing environment to a data sharing environment, you should consider the following points:

- When you are specifying a recovery, you can use TORBA or TOLOGPOINT keywords interchangeably.
  
  If you are specifying a point in time before the conversion to a data sharing environment, you should specify an RBA value. If you are specifying a point in time after the conversion to a data sharing environment, you should specify an LRSN value.

- If BSDS passwords are used in a data sharing environment, NGT Recover will work only if all of the passwords for the group are the same.

  OPNDB2ID will work under data sharing only if all Resource Access Control Facility (RACF) IDs for the members of the group are the same. The BSDS authorizations must also be the same.

- NGT Recover supports recovery in a data sharing environment by using log records written before data sharing was enabled. You can recover table spaces by using image copies that were made before data sharing was enabled.

- The DB2 RECOVER utility does not support recovery after data sharing is disabled if the recovery requires log records that were created while data sharing was active (For more information, see DB2 for z/OS Data Sharing: Planning and Administration).

  NGT Recover does support such recovery requests. However, if you re-enable data sharing and any table spaces or indexes have not been copied since before data sharing was originally enabled, BMC recommends that you copy them. Failure to do so may render those objects unrecoverable.

## Forward recovery of versioned index spaces

NGT Recover supports the forward recovery of a versioned index.
Versioning occurs when the attributes of an index column have been altered. NGT Recover needs information from the header page and the directory page to recover such an index space.

- The header page is always the very first page of any partition or the very first piece of a nonpartitioned index (NPI).

- Page 4 will always be a directory page. If the index has been versioned several time, there may be more directory pages throughout the index.

  BMC strongly recommends that you rebuild or REORG indexes with more than one directory page if you are creating image copies on a DB2 subsystem prior to DB2 Version 10. Otherwise, the recovery may fail. With DB2 Version 10, the directory pages will be written at the beginning of the image copy, insuring that the recovery will be successful.

With image copies made prior to DB2 Version 10, if a multi-data-set, nonpartitioned index contains a versioned column, the very first piece of the NPI must be included in any request to recover the NPI. NGT Recover must have at least a header page to process the recovery correctly. If the NPI was altered prior to migrating, and the index was not rebuilt or reorganized, the recovery will fail with the following error if the directory page is contained in a piece that is not included in the recovery.

BMC4D855 CANNOT OBTAIN DIRECTORY PAGE INFORMATION. REQUESTED RECOVERY IS NOT POSSIBLE.

In the case of a versioned index, NGT Recover reorders the recovery request to use the correct order of DSNUM. For example, NGT Recover accepts the following command statements, but reorders the request to process DSNUM 1 first:

RECOVER INDEX (I1) DSNUM 5
RECOVER INDEX (I1) DSNUM 1

Supporting real-time statistics in NGT Recover

NGT Recover enables real-time statistics as follows:

- NGT Recover resets the COPY-related counters, even during a recovery to current, if it is making an output image copy during a recovery.

- NGT Recover does not update any real-time statistics during an OUTCOPY ONLY execution.

The following table provides the columns in SYSIBM.SYSTABLESPACESTATS1 and SYSIBM.SYSINDEXSPACESTATS that the NGT Recover RECOVER command updates.
Table 26: Columns updated by the NGT Recover RECOVER command for real-time statistics

<table>
<thead>
<tr>
<th>Column name</th>
<th>Updated value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATESTATSTIME</td>
<td>Timestamp of update</td>
</tr>
<tr>
<td>COPYLASTTIME</td>
<td>Timestamp</td>
</tr>
<tr>
<td>COPYUPDATEDPAGES</td>
<td>0</td>
</tr>
<tr>
<td>COPYCHANGES</td>
<td>0</td>
</tr>
<tr>
<td>COPYUPDATELSRN</td>
<td>Null</td>
</tr>
<tr>
<td>COPYUPDATETIME</td>
<td>Null</td>
</tr>
</tbody>
</table>

This field is updated by the NGT Recover RECOVER command only when an output image copy is made during a recovery. Because making an output image copy during recovery is not an option with the DB2 RECOVER utility, the DB2 RECOVER utility never updates these fields.

The DB2 REBUILD utility updates some reorganization-related statistics in SYSIBM.INDEXSPACESTATS, as well as the REBUILDLASTTIME column. The following table summarizes the support for updating these statistics when you use the NGT Recover REBUILD command.

Table 27: Columns updated by the NGT Recover REBUILD command for real-time statistics

<table>
<thead>
<tr>
<th>Column name</th>
<th>Updated value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATESTATSTIME</td>
<td>Timestamp of update</td>
</tr>
<tr>
<td>REBUILDLASTTIME</td>
<td>Timestamp</td>
</tr>
<tr>
<td>REORGINserts</td>
<td>0</td>
</tr>
<tr>
<td>REORGDELETES</td>
<td>0</td>
</tr>
<tr>
<td>REORGAPPENDINSERT</td>
<td>0</td>
</tr>
<tr>
<td>REORGPSEUDODELETES</td>
<td>0</td>
</tr>
<tr>
<td>REORGMASSDELETE</td>
<td>0</td>
</tr>
<tr>
<td>REORCLEAFNEAR</td>
<td>0</td>
</tr>
<tr>
<td>REORCLEAFFAR</td>
<td>0</td>
</tr>
<tr>
<td>REORGNUMLEVELS</td>
<td>0</td>
</tr>
</tbody>
</table>

Recovering altered spaces

NGT Recover supports online schema evolution, including recovering spaces that have had:
Added partitions

If you add a partition with an ALTER statement or if a partition is automatically added with a partition-by-growth (PBG) universal space, you can perform the following recoveries with NGT Recover:

- You can recover the partition to the current time.
- You can recover the partition to a point in time after the partition was added.
- You can recover spaces to a point in time before the partition existed (prior to the ALTER ADD PARTITION), and NGT Recover creates the partition with no data.

If a point-in-time recovery causes a partition-by-growth table space to shrink and leave the space empty, the underlying data set will contain a header and space map. The number of partitions will not be decreased in the catalog.

**Note**

For universal table spaces, if you use the LASTQUIESCE option and the quiesce is prior to the ALTER ADD PART, the recovery will fail. You must specify a hard-coded RBA. If the partition was added for a partition-by-growth space, you can use the LASTQUIESCE option.

When you are performing a drop recovery or a migration of a range-partitioned table space, the number of source partitions must match the target.

- For a DROPRECOVERY, IMPORT, or INCOPY request of PBG universal spaces where the number of source data sets differs from the number of target data sets, NGT Recover determines the number of source data sets and acts accordingly.

- For a MIGRATE or INDEP OUTSPACE request, if the number of source data sets is greater than the number of target data sets, the request fails. If the number of target is greater than the number of source data sets, the unused target data sets is reset with a header page and space map.

Rotated partitions

If you rotate (ALTER ADD ROTATE) a partition, you can perform the following recoveries with NGT Recover:
You can recover the partition to the current time.

You can recover the partition to a point in time after the rotate.

You cannot recover table or index space partitions to a point in time before the partition was rotated.

---

**Inserted partitions**

If you use ALTER ADD PARTITION to insert a new partition in the middle of the table in a range-partitioned table space, you can perform the following recoveries with NGT Recover:

- You can recover the partition to the current time.
- You can recover the partition to a point in time (PIT) after the partition insertion.

You *cannot* recover a table space or specific partition to a PIT before the new partition insertion.

---

**Dropped partitions**

If trailing partitions are removed from a PBG table space by a REORG on a subsystem where DSNZPARM parameter REORG_DROP_PBG_PARTS is enabled, you can perform the following recoveries with NGT Recover:

- You can recover the remaining partitions to the current time.
- You can recover the remaining partitions to a point in time after the trailing partitions were removed.

You *cannot* recover table or index space partitions to a point in time before the partitions were removed.

---

**Changed limit keys**

If you change the limit keys on a partitioned space, the affected partitions are placed in REORP status and you cannot use the affected partitions until you run a REORG on those partitions.

DB2 added the REBALANCE feature to the REORG utility that changes limit keys and does a REORG of the specified partitions in one step, which avoids having the spaces put in REORP status.
However, starting with DB2 Version 11, the change of limit keys for range-partitioned table is treated as pending change. In this case, the affected partitions are placed in AREOR status and are recorded in the SYSIBM.SYSPENDINGDDL DB2 catalog table.

If you recover to a point in time that precedes the change, you need to reorganize the spaces, and they are placed in REORP status.

Materialized pending changes

NGT Recover allows you to recover to a point in time (PIT) before a reorganization that materialized some types of online pending changes.

You can use this feature only with a range-partitioned or partition-by-growth universal table space (PBR UTS or PBG UTS), an XML space, or a LOB space. You can perform a PIT recovery over the following pending changes:

- PBR UTS
  - DSSIZE
  - PGSIZE
  - SEGSIZE
  - MEMBER CLUSTER

- PBG UTS
  - DSSIZE
  - PGSIZE
  - SEGSIZE
  - MEMBER CLUSTER

- XML auxiliary space
  - DSSIZE
  - SEGSIZE

- LOB auxiliary space
  - DSSIZE
  - PGSIZE

The PIT recovery sets the table space in REORP status and inserts a row into the SYSIBM.SYSPENDINGDDL DB2 catalog table. You must reorganize the entire table space to complete the PIT recovery and remove the row from the SYSIBM.SYSPENDINGDDL DB2 catalog table.
Following are restrictions that apply when you recover to a PIT that precedes materializing the pending definition changes:

- You must recover the entire table space. You cannot recover specific partitions.
- You cannot recover indexes.
- There must not be any current pending changes.
- The table space attributes cannot be in a CLONE relationship.
- You cannot perform recovery to a subsequent and different PIT until after the reorganization that completes the recovery.

**Note**
After executing a reorganization to materialize a DROP COLUMN pending change, you cannot perform a PIT recovery to a point before the materializing reorganization.

## Recovering encrypted copies

The use of encryption protects sensitive company information and prevents security failures.

NGT Recover support for encryption allows you to recover using image copies that are protected from unauthorized access to the sensitive information. For the recovery, NGT Recover uses full and incremental encrypted copies made by NGT Copy. For information about making encrypted copies, see the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*.

**Note**
Encryption is a feature of the BMC Recovery Management for DB2 solution and requires a valid Recovery Management solution password.

Encryption is based on standard secret key encryption algorithms. In NGT Copy, you select encryption based on one of three following standard algorithms:

- The ANSI Data Encryption Algorithm (DEA) with a 64-bit key
  This is the default algorithm. This algorithm is also known as the U.S. National Institute of Science and Technology Data Encryption Standard (DES).
- The Triple Data Encryption Standard (TDES) with a 128-bit key
- The Advanced Encryption Standard (AES) with a 128-bit key
NGT Copy supports encryption of plaintext image copies or decryption of cipher
text image copies. *Plaintext or clear text is data in normal, readable form.* (NGT
Copy standard image copies are plaintext.) *Encrypted text or cipher text is data that
has been converted to mask its meaning from an unauthorized recipient.*

**Requirements for encryption**

To specify that you want to recover by using encrypted copies made by NGT Copy,
you must:

- Run NGT Recover on a processor that supports encryption
- For a *non-registered encrypted image copy*, use the ENCRYPTED option and
  TIMESTAMP option in the INCOPY specification of the RECOVER command
  ("INCOPY specification" on page 182)

  **Note**
  
  NGT Recover finds encrypted copies registered in BMCXCOPY without the use
  of the syntax options.

- Use the KEYDSNAM installation option to specify your key data set name
  ("KEYDSNAM= keyDataSetName" on page 586)
- Create and maintain the key data set ("Key data set" on page 507)
- Have the Recovery Management *for DB2* solution and use a valid Recovery
  Management password

**Key data set**

Support for encryption in NGT Recover relies on a user-created and maintained data
set, called the key data set.

The key data set contains essential encryption key information. NGT Copy requires
the key data set to make encrypted copies. NGT Recover requires the same or an
identical key data set used by NGT Copy to recover by using encrypted copies.

**Key data set requirements**

You must perform the following tasks for the key data set:
Create the key data set.

NGT Recover requires that the key data set be a fixed or fixed block physical sequential data set with a logical record size (LRECL) of 80. NGT Recover requirements for the contents of the data set are specified in “Key data set contents” on page 509. Any variation from these requirements could prevent NGT Recover from recovering the encrypted image copy.

**Note**

BMC recommends that you use the same key data set that NGT Copy used to encrypt the image copy or a duplicate of that NGT Copy key data set. A duplicate data set is required if you are migrating data from one system to another, and the key data set is not shared between the systems or you are recovering data at a remote site.

Identify the key data set to NGT Recover.

The KEYDSNAM installation option (“KEYDSNAM= keyDataSetName” on page 586) specifies the key data set name. After you specify the key data set name, NGT Recover dynamically allocates the data set when it is needed.

If NGT Recover attempts to recover an encrypted image copy and you have not specified the key data set name in the installation options, NGT Recover issues the following message:

```
BMC40020I ENCRYPTION KEY DATA SET NOT SPECIFIED IN OPTIONS MODULE
```

If NGT Recover attempts to recover an encrypted image copy and the key data set is not cataloged, NGT Recover issues the following message:

```
BMC40020I ENCRYPTION KEY DATA SET NOT CATALOGED
```

Maintain the key data set.

Periodically, you may want to change encryption keys. You cannot edit the key data set while any utility that is using the key data set is inflight. You need to schedule time to maintain the data set. You must take care when you maintain the data set because incorrect entries in the data set might prevent NGT Copy from encrypting your image copies or prevent NGT Recover from recovering from a previously encrypted image copy.

Provide appropriate security for the key data set to protect it from unauthorized access.

**Note**

If a non-registered encrypted image copy is used in the INCOPY clause and the registration timestamp is not known, a timestamp with a value greater than the timestamp of the desired encryption key, but less than subsequent encryption keys can be used in place of the registration timestamp. Due to security, the administrator of the key data set will need to determine this value.

Maintain backups of your key data set either with DFSMShsm or some other facility.
Key data set contents

The key data set contains one or more rows of 80 characters per row.

NGT Recover ignores any characters in columns 72 through 80.

Each row contains:

- One encryption key
- A corresponding timestamp
- An optional encryption algorithm identifier
- An optional comment

These fields are separated by one or more blank characters. The first character of the comment is an asterisk. Rows are ordered in the data set by timestamp with the most recent timestamp first. The current key is the key in the first row. The format of the key data set row is:

<table>
<thead>
<tr>
<th>Key value</th>
<th>Timestamp</th>
<th>Encryption algorithm ID</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'0ABCDEF123456789FEDCBA00011111'</td>
<td>2012-11-23-12-00</td>
<td>128 bit DES encryption</td>
<td></td>
</tr>
<tr>
<td>X'123456789ABCDEF1'</td>
<td>2012-08-23-11-10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X'723DE6789000DEF1'</td>
<td>2011-12-12-16-40</td>
<td>DES</td>
<td>64 bit DES encryption</td>
</tr>
<tr>
<td>X'723DE6789000DEF1723DE6789000DEF1'</td>
<td>2011-12-12-14-00</td>
<td>AES</td>
<td>128 bit AES encryption</td>
</tr>
<tr>
<td>X'F1F2F3F4F5F6F7F8'</td>
<td>2011-01-01-12-00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NGT Recover uses the contents of the key data set to determine a key value for decryption of image copies. NGT Recover uses either the first key whose timestamp is less than the timestamp in BMCXCOPY or the timestamp specified in the INCOPY specification.

Encrypted image copies are registered in BMCXCOPY. As with SYSCOPY registration, BMCXCOPY registration includes a timestamp specifying when the copy was registered. NGT Recover uses this timestamp to find the correct key value in the key data set. For more information about the registration of encrypted copies, see “Registration for plaintext image copies” on page 512.

For example, if NGT Recover selected an image copy for a recovery from BMCXCOPY with a timestamp of 2011-02-12-10.00, the encryption key and DES algorithm in the third row in the example key data set above is selected.

Key value

NGT Recover supports both 64-bit and 128-bit keys. (See “Encryption algorithm identifier” on page 510.) The key data set can contain either or both key sizes. The
key value is a clear key represented in the key data set as a string of 16 or 32 hexadecimal digits in the following format:

X'dd...'

The X and the quotes are required. The X must occur in the first column and be upper case.

**Timestamp**

The date, hour, and minute string uses following formats:

```
yyyy-mm-dd-hh-mm
```

or

```
yyyy-mm-dd-hh.mm
```

The values are decimal numbers and are padded on the left with a zero if necessary. The timestamp must be separated from the key value by at least one blank space.

**Encryption algorithm identifier**

An encryption algorithm identifier is optional in the key data set. The encryption algorithm identifiers supported are

- DES for Data Encryption Standard (for 64-bit keys)
- DES for Triple Data Encryption Standard (for 128-bit keys)
- AES for Advanced Encryption Standard (requires 128-bit keys)

The encryption algorithm identifier defaults to DES if no identifier is provided. If you provide an identifier, you must separate it from the timestamp by at least one blank. NGT Recover distinguishes between the two varieties of DES based on the length of the key (64-bit or 128-bit).

**Comments**

Comments are optional in the key data set. A comment begins with an asterisk that is separated from the preceding field by at least one blank.

**Key data set management**

The security of the encrypted NGT Copy image copies and the ability of authorized individuals to use NGT Recover to recover DB2 spaces using these image copies depends on the careful management of the key data set.
BMC recommends that you develop a simple and well-documented mechanism to manage key data sets.

BMC recommends that you maintain one key data set shared by all systems with access to the data set. Multiple distinct key data sets create difficulty with key data set management because you must ensure that the key data set that is used to encrypt an image copy is also used for recovery with that encrypted image copy.

Consider all of the following items as you manage your key data set:

- **Protect the key data set on the local system and duplicates on remote systems against unauthorized access.**
  Most attempts to access encrypted data occur as unauthorized access to the key data set. You should protect the key data set against unauthorized access during shipping with either a secret key or public key encryption. If key data set is not encrypted during shipping, it should never be shipped under the same cover as the encrypted image copies.

- **If you plan to use encrypted image copies at your disaster recovery site, be sure that the processor at the site supports encryption.**
  Remote disaster recovery sites may require a duplicate key data set for recovery purposes.

- **Because the timestamps that are used for recovery are taken from the BMCXCOPY table, a change in time zones between the site where NGT Copy made the encrypted image copies and the disaster recovery site will not affect recovery.**
  The possibility exists, however, that a time zone change might invalidate a key data set for creating image copies at the remote site. If this is the case, you will need a new key data set with local times for generating encrypted image copies at the remote site.

- **Limit updating of the key data set to authorized individuals.**
  Generating a new current key by inserting a new first row in the key data set limits the amount of data exposed if the current key is compromised. Do not modify existing rows in the key data set because image copies may exist that will require the keys for recovery. It is important that duplicate key data sets on remote systems also contain this new row, and that backups of the key data set be immediately created on all systems.

- **Once image copies encrypted by a key are no longer referenced in the local and remote BMCXCOPY tables and will not be used by NGT Recover for recovery as a non-registered image copy, the key is no longer needed by NGT Copy, NGT Recover, or Log Master and you can eliminate the key.**
  Key destruction steps are:
1. Delete backups of the current key data set on both the local and remote systems.

2. Remove the row containing the key from the local key data set and duplicate key data sets on remote systems.
   Never remove a row from the key data set unless it is the last row in the data set.

3. Create backups of the new key data set on the local and remote systems.

If a key data set is lost or corrupted and not recoverable, you can gain emergency access to the current key data set with a technique called *key escrow*. Once you have created or updated a key data set, the contents are divided into two or more partial key data sets so that no one data set is sufficient to decrypt an image copy. Each partial key data set is sent to different trusted agent. In the event of an emergency, you can retrieve and reassemble the partial data sets.

### Registration of copies

The topic describes the registration of encrypted copies.

#### Registration for encrypted image copies

Because the encrypted image copies produced by NGT Copy are non-standard, encrypted image copies are registered in the BMCXCOPY table.

An STYPE value of *e* indicates that the image copy is encrypted. In a recovery that uses these encrypted image copies that are registered in BMCXCOPY, you must use NGT Recover.

#### Registration for plaintext image copies

Plaintext full image copies are registered in SYSCOPY.

Instant Snapshot copies and certain index space copies are exceptions and are registered according to the rules for BMCXCOPY. (For more information, see the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*.)

A plaintext incremental is registered in SYSCOPY if the most recent primary full copy of the same site type is also plaintext. If the most recent primary full copy of the same site type is encrypted, the incremental is registered in BMCXCOPY.
You can use NGT Recover or the DB2 RECOVER utility to recover using plaintext copies that are registered in SYSCOPY. But you must use NGT Recover for recovery if the plaintext image copies are registered in BMCXCOPY.

Handling DB2 versioning information

When you move objects that contain system pages from one DB2 subsystem to another DB2 subsystem, the version information on the target DB2 subsystem must match the version information on the source DB2 subsystem. If the version information does not match, you cannot access the data on the target DB2 subsystem.

If your version information does not match, use the following steps to move objects to another DB2 subsystem and to ensure that the version information matches.

To move objects to another DB2 subsystem

1. Ensure that the object definitions on the source and target DB2 subsystems are the same.
   For a table space, each table must have the same number of columns, and each column must be the same data type.
   
   **Tip**
   Use the same ALTER TABLE statement on both the source and target objects.

2. If you are copying indexes that have not been altered in DB2, check the SYSIBM.SYSINDEXES catalog table on both subsystems to ensure that the value in both the CURRENT_VERSION column and the OLDEST_VERSION column is 0.
   
   **Tip**
   Ensure that you do an insert after the last ALTER to a table to force the creation of a system page, or use NGT Copy with the GENSYSPPAGES option to make the copy. If you are copying indexes and an index was altered, you may need to REBUILD the index to update the index directory page with the latest version before making the copy.

3. Run NGT Recover on the target subsystem with the OBIDXLAT option. On the control statement, specify the proper mapping of table database object identifiers (OBIDs) for the table space or index from the source to the target subsystem.
Note
If you need to rebuild indexes on your target system, do not do so until after you have completed Step 4 on page 514.

4 Run DB2 REPAIR VERSIONS on the object on the target subsystem or specify the NGT Recover UPDATE VERSIONS option in Step 3 on page 513.

This action updates the version numbers in the target system’s catalog using the version numbers in the system pages or index directory pages that have been laid down from the source system.

For table spaces, the utility updates the following columns:

- OLDEST_VERSION and CURRENT_VERSION in SYSTABLEPART
- VERSION in SYSTABLES
- OLDEST_VERSION and CURRENT_VERSION in SYSTABLESPACE

For indexes, the utility updates OLDEST_VERSION and CURRENT_VERSION in SYSINDEXES. DB2 uses the following formulas to update these columns in both SYSTABLEPART and SYSINDEXES:

\[
\begin{align*}
\text{CURRENT\_VERSION} &= \max(\text{target.CURRENT\_VERSION}, \text{source.CURRENT\_VERSION}) \\
\text{OLDEST\_VERSION} &= \min(\text{target.OLDEST\_VERSION}, \text{source.OLDEST\_VERSION})
\end{align*}
\]

5 If you need to rebuild indexes on your target system, you can do so now.

Recovering cloned objects

NGT Recover supports the recovery of cloned tables.

Use the CLONE option on the following NGT Recover commands to recover with clone objects:

- RECOVER
- REBUILD
- RECOVER UNLOADKEYS
- RECOVER BUILDINDEX
- LOGSCAN

When you are creating an output copy or a model DCB, you can use the symbolic variable &INST for the instance number (Table 25 on page 495). Valid values are 1 and 2.
A clone table is a table with the exact same attributes as an existing table, which is called the base table. The clone is created in the same table space as the base table, is structurally identical to the base table in every way and has the same indexes, triggers, and LOB objects. You can only create clone tables in a universal table space that is managed by DB2.

The base and clone table data and their index data reside in different data sets. For example:

- `catName.DSNDBx.dbName.psName.x0001.A001`
  This data set contains instance number 1 (the 1 of x0001).
- `catName.DSNDBx.dbName.psName.x0002.A001`
  This data set contains instance number 2 (the 2 of x0001).

DB2 handles log records and log ranges for both base and clone table objects. The base and clone objects are differentiated by differences in the PSID value. The high order bit of the PSID is used to refer to a particular data set instance number as follows:

- A high order bit value of 0 indicates instance number 1
- A high order bit value of 1 indicates instance number 2

Other than the high order bit, the PSID numbers are the same for both a base object and its clone.

NGT Recover support for clone objects includes the following limitations:

- You cannot specify the same object with and without the CLONE option in the same SYSIN data set.
  NGT Recover issues an error message if you specify a recovery command for the base and clone instance of the same object in the same SYSIN data set.

- All related objects (indexes and underlying table space, LOB objects) should have the same CLONE specification in the same step.
  NGT Recover issues an error message if the CLONE specification does not match for the related objects.

- You cannot recover a clone or base object to a point prior to the last EXCHANGE statement.
Recovering XML objects

NGT Recover supports the recovery of XML objects. NGT Recover also supports REBUILD requests for NodeID indexes.

When an XML column is created, DB2 implicitly creates the following XML objects to support the XML column:

- DocID column in the base table—created on the base table and uniquely represents each row.
- DocID index
- XML table
- XML table space containing the XML table
- NodeID index on the XML table

For a point-in-time recovery, you must recover the following XML-related objects to the same point:

- Base table—contains the base table, where the logical XML column is stored
- XML table space—contains the XML auxiliary table, where the data is physically stored
- NodeID index on the XML table
- DocID index

**Note**

In the CREATE statement and in the DB2 catalog, XML indexes are defined on the XML base table. However, the XML indexes are actually on the XML table (table type ‘P’).

Because of this, the IBM RECOVER utility does not process XML indexes in INDEX(ALL) TABLESPACE baseTableSpace requests. In order to process XML indexes with an INDEX(ALL) request, you must specify TABLESPACE XMLTableSpace.

For XML indexes, NGT Recover behavior is the same as that of IBM.

The following restriction applies to the recovery of an XML object:

- A point-in-time recovery can always convert an XML object from extended to basic RBA format. However, the point-in-time recovery for an XML table space cannot convert from basic to extended format.
Recovering compressed indexes

NGT Recover provides native support for recovering copies of compressed indexes made by DB2 or NGT Copy. (NGT Copy provides native support for copying compressed indexes.)

Additionally, NGT Recover provides native support for

- Backout recovery of compressed indexes
- Recovery of encrypted compressed indexes
- Recovery of cabinet copies of compressed indexes
- Optimized merge recovery of compressed indexes
- Rebuilding compressed indexes

Two methods are available to make copies of compressed indexes. Each method has advantages and disadvantages.

- Make copies of compressed indexes by expanding them

  Both the DB2 COPY utility and NGT Copy can make copies of compressed indexes by reading the compressed indexes directly from disk and expanding them before writing the image copy. This method has the advantage that the copy is registered in SYSCOPY (and, in the case of NGT Copy, the copy is compatible with those produced by the DB2 COPY utility).

  Some disadvantages of this method are that it takes time to expand the pages and the resulting image copies are larger than necessary. Also, NGT Copy copy techniques, such as Instant Snapshot copies and Online Consistent Copies, are not supported. (NGT Copy will make unexpanded copies if you request an expanded copy.)

- Make copies of compressed indexes without expanding them

  NGT Copy can make copies of the compressed indexes without expanding them. NGT Copy registers these copies in the BMCXCOPY table. This method has the advantages that making the copies is faster and the copies are smaller than when copies are made of expanded indexes. Also, all NGT Copy copy techniques, such as Instant Snapshots, Online Consistent Copy, encrypted copies, and cabinet copies, are supported.

  When you make copies of compressed indexes without expanding them, the copy is not compatible with the DB2 utilities and NGT Recover is required to use these copies for recovery.

NGT Recover supports both expanded and unexpanded copies of compressed indexes, and automatically determines the type of the copies that are available.
When you use the OUTCOPY option in NGT Recover, copies of compressed indexes must be unexpanded.

When you use NGT Copy, the IXEXPAND option indicates which method you want to use to create copies of compressed indexes. Valid values are AUTO, YES, and NO. The default value is AUTO. IXEXPAND AUTO specifies that NGT Copy will make unexpanded copies of compressed indexes if you are running with a Recovery Management solution password (and consequently have NGT Recover to recover using the unexpanded copies). If you are not running Recovery Management, NGT Copy will make expanded copies for compatibility with IBM. IXEXPAND NO specifies that compressed index copies are unexpanded. IXEXPAND YES specifies that compressed index copies are expanded. For more information on the IXEXPAND option, see the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*.

If you do not have the Recovery Management solution, but you do have the NGT Recover and NGT Copy products, you should consider changing the value of the NGT Copy IXEXPAND installation option to NO to achieve the benefits of copying the compressed indexes without expansion.

### Using BMC RECOVERY MANAGER groups

You can set up groups in the BMC RECOVERY MANAGER for DB2 product. NGT Recover allows you to identify both table spaces and index spaces in groups for processing by NGT Recover.

Using dynamic grouping, NGT Recover reads the objects in the group each time you run the NGT Recover job, so objects may be added or removed from the group between NGT Recover job runs.

NGT Recover does not read RECOVERY MANAGER recovery options for the group. If RECOVERY MANAGER recovery options change, you must run RECOVERY MANAGER to pick up the new options and generate the control cards.

Once the group is defined using RECOVERY MANAGER, NGT Recover uses the RECOVERY MANAGER repository and the BMC Common DB2 repository to identify the objects in the group. NGT Recover uses the objects directly from the repository tables. The objects in the group are kept up to date with dynamic grouping.
Dynamic grouping resolves the table space and index object names for inclusion with the NGT Recover commands that support group object types.

NGT Recover provides the OBJECTSET syntax to specify a group. The long range plan is that OBJECTSET will be used as common syntax by other BMC utilities for DB2. You can use OBJECTSET with the following NGT Recover commands:

- IMPORT
- LOGSCAN
- MIGRATE
- RECOVER
- REBUILD INDEX
- SIMRCVR
- SIMBLD INDEX

For more information, see the syntax descriptions for each command in “NGT Recover syntax” on page 83, and specifically, “OBJECTSET specification” on page 177.

Following are some examples of the use of the OBJECTSET syntax:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Objects copied</th>
</tr>
</thead>
</table>
| RECOVER OBJECTSET creator. name | Recovers all objects in the object set—table spaces and indexes  
Note that INDEXES YES is not valid with this specification. |
| RECOVER TABLESPACE OBJECTSET creator. name | Recovers only table spaces in the object set except for those table spaces named in the EXCLUDE statement |
| RECOVER TABLESPACE OBJECTSET creator. name INDEX YES | Recovers table spaces in the object set along with their associated indexes, regardless of whether the indexes are included in the group  
You can use the EXCLUDE statement to ignore one or more table spaces and their associated indexes. |
| RECOVER INDEX OBJECTSET creator. name | Recovers only indexes in the object set except for those indexes named in the EXCLUDE statement |
| REBUILD INDEX OBJECTSET creator. name | Rebuilds all indexes in the object set except for those indexes named in the EXCLUDE statement |
| LOGSCAN OBJECTSET creator. name | Log scan processing of all objects in the object set—table spaces and indexes |
| LOGSCAN TABLESPACE OBJECTSET creator. name | Log scan processing for only table spaces in the object set |
| LOGSCAN TABLESPACE OBJECTSET creator. name INDEX YES | Log scan processing for table spaces in the object set along with their associated indexes, regardless of whether the indexes are included in the group |
### Syntax

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Objects copied</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGSCAN INDEX OBJECTSET <em>creator_name</em></td>
<td>Log scan processing for only indexes in the object set</td>
</tr>
<tr>
<td>a SIMRCVR is also valid.</td>
<td></td>
</tr>
<tr>
<td>b SIMRBLD is also valid.</td>
<td></td>
</tr>
</tbody>
</table>

#### Note

DSNUM cannot be used with OBJECTSET. The DSNUM used for each object is its DSNUM in the repository.

When you use OBJECTSET, if NGT Recover tries to recover a group of objects and more than 10 of the objects are unrecoverable, NGT Recover issues an error for each object that is unrecoverable and discontinues further processing of the group. This behavior is based on the ERRCONT option that has a default value of 10. To avoid this behavior, set the ON ERROR CONTINUE option with a value that will exceed the possible number of errors in the group, and the recovery will complete.

For more information about grouping in RECOVERY MANAGER, see the *RECOVERY MANAGER for DB2 User Guide*.

**Using MIGRATE and IMPORT for data movement**

With the Copy Migration feature of the Recovery Management solution, you can use the NGT Recover MIGRATE and IMPORT commands to migrate a copy or set of copies to another DB2 subsystem.

The MIGRATE and IMPORT commands work with a sequential file created by the NGT Copy EXPORT command. The sequential file contains BMCXCOPY and SYSCOPY table information about all selected table spaces. The NGT Recover MIGRATE and IMPORT commands use the file to import migrated copies into the target location.

EXPORT, MIGRATE, and IMPORT make the task of moving the data between DB2 subsystems simpler and more accurate, and support all NGT Copy formats (including cabinet copies).

If your migration uses the MIGRATE command, the NGT Copy EXPORT runs on the target system and the NGT Recover MIGRATE runs on the source system. NGT Recover uses the OBIDs from the EXPORT. Optionally, log from the source system can be used with the MIGRATE command.
The EXPORT file replaces the need for INCOPY and OBIDXLAT syntax with NGT Recover. Records written to the file identify the correct data sets to use and how to translate the OBIDs from the source subsystem to the target subsystem.

This copy migration feature requires one of the following valid passwords:

- A Recovery Management solution password
- A Database Administration solution password

For more information about the NGT Copy EXPORT command, see the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*.

The BMC CHANGE MANAGER product also has a data migration function that handles DDL definition and OBID resolution. CHANGE MANAGER also uses the Copy Migration feature (EXPORT and IMPORT only, not MIGRATE) of NGT Copy and NGT Recover. For more information, see the CHANGE MANAGER product documentation.

The benefits of data migration with the NGT Copy EXPORT and NGT Recover MIGRATE/IMPORT feature include:

- Supports importing copies that use BMC proprietary formats
- Does not require that you manage data sets to keep up with current copies
- Does not require that you know the OBIDs from the source system
- Imports applications and object sets as a single unit
- Does not require the import of unchanged objects
- Supports the use of older EXPORT files to back-date imports
- Provides a less error prone, and therefore, more accurate data migration (for example, less likely to use the wrong data set name or OBIDs).

If you use CHANGE MANAGER with the Copy Migration feature (EXPORT and IMPORT commands only, not the MIGRATE command), you achieve the following benefits:

- Simplifies the data movement process
- Requires less JCL, and therefore, less JCL management
- Supports DSNUM values other than 0
- Supports multiple imports and multitasking
- Ability to use migration resources stored in BMCXCOPY, such as cabinet and OCC image copies
The NGT Copy EXPORT file provides a method for transferring all types of NGT Copy and IBM copies between DB2 subsystems.

The EXPORT file contains information to streamline the import recovery process with a minimum of user-supplied knowledge and intervention. The MIGRATE or IMPORT command of NGT Recover is used to complete the transfer and integration of data using the copies.

The latest SYSCOPY and BMCXCOPY rows and metadata for the selected spaces are written to the designated sequential file. The metadata describes each object with information needed to translate OBIDs on the target system and to do checks during importing.

**Note**

EXPORT requires that the image copy is consistent to a point in time due to IMPORT processing on the target system done as TOCOPY. No log files are applied at the target. As the copies are selected for exporting, an attempt is made to verify that they are for the same consistent points.

If NGT Copy finds an invalid image copy, NGT Copy issues a message that indicates no copy was found. NGT Copy assumes the EXPORT is for migration and continues processing. If a valid image copy is found, NGT Copy issues a message that identifies the spaces that qualify for exporting.

The EXPOUT option on the NGT Copy OUTPUT command indicates that NGT Copy will create an EXPORT file for migration.

**NGT Recover MIGRATE and IMPORT commands**

To initiate the import process, at a minimum you must specify the MIGRATE or IMPORT command with a USING clause. If table name ambiguity exists, you can add the CHANGE ... FROM ... TO clause. When all the tables on each target space match all the tables on one and only one space in the NGT Copy EXPORT file, you do not need to specify the CHANGE ... FROM ... TO clause. Additionally, a SYNC option on the IMPORT command allows you to specify whether you want to migrate all spaces or only changed spaces.

For NGT Recover MIGRATE and IMPORT command syntax, see “MIGRATE” on page 270 and “IMPORT” on page 290, respectively.
Limitations

The EXPORT/MIGRATE or IMPORT copy migration feature has the following limitations:

- No support for incremental copies
- No support for copies of the DB2 catalog
- No support for specifying DSNUM
  
  EXPORT looks for the most recent copy. If the copy is a DSNUM 0 copy, EXPORT uses that copy. If the copy is a DSNUM \( n \) copy, EXPORT looks for DSNUM \( n \) copies for all partitions or data sets.

- For IMPORT, NGT Recover recovers indexes that are included in the IMPORT file and rebuilds other indexes, unless you specify INDEXES NO. If the source and target table spaces are not at the same version, NGT Recover will not rebuild the indexes. You will need to run REPAIR VERSIONS or use the UPDATE VERSIONS option, and then rebuild the indexes in a separate step.
Recovering a dropped table space or table

This chapter provides an overview and examples for the recovery of a dropped table space or table.

Overview of recovering dropped table spaces or tables

The following considerations apply to drop recovery:

- Drop recovery for indexes is not supported.

- When you are performing a drop recovery for a range-partitioned table space, the number of partitions must match. For the drop recovery of a partition-by-growth (PBG) universal spaces where the number of source data sets differs from the number of target data sets, NGT Recover determines the number of source data sets and acts accordingly.

- If the dropped space was versioned, you must recreate the space with the same versions, or in some cases, you can use DB2 REPAIR VERSIONS.

  — If the dropped space has been reorganized after the first alter, you can recreate the space to match the latest version of the space, recover the space, and then run DB2 REPAIR VERSIONS. You cannot rebuild indexes until after you run DB2 REPAIR VERSIONS.

  — If the dropped space has not been reorganized, you must recreate the space as it existed originally, before any alters, perform the alters to create the same versions that the dropped space had, and then recover the space.

- For more information, see “Handling DB2 versioning information” on page 513.
The following figure shows how you can use the NGT Recover product to recover a dropped table space.

**Figure 144: NGT Recover drop recovery process**

To recover a dropped table space, you must first re-create the DB2 table space and index structures. To simplify the task of re-creating objects, you can use the BMC CATALOG MANAGER, ALTER, or CHANGE MANAGER for DB2 products.

NGT Recover rebuilds the data in the table space as it existed at the time of the drop by

- Using the OBIDXLAT clause to translate the old OBIDs
- Using the INCOPY option for image copies that are now unregistered as input
- Using the NOSYSLGRNG option to bypass the SYSLGRNX table when determining which log to scan

**Example 1: Recovering a dropped table space when image copies are available**

A DBA new to your company has accidentally dropped a production table space instead of the test table space that he intended to drop.

You have been assigned to recover this table space to the point in time at which it was dropped. You have already identified the most recent image copy data set.
To recover a dropped table space when image copies are available

1 Obtain data definitions.

Obtain the definitions of the table space and table that you want to recover. Also, obtain definitions for the indexes because they have also been dropped. You cannot change any characteristics of the table space or table that affect their internal structure.

Note

You can obtain the object definitions by using a tool such as one of the following BMC products: CATALOG MANAGER for DB2, ALTER for DB2, or CHANGE MANAGER for DB2. (If you use ALTER for DB2, it can systematically change object names using a name mask in a migrate profile, which could be helpful if many table spaces are involved.)

The data definition language (DDL) for this example is shown in the following figure.

Figure 145: Example DDL for the dropped table space

```sql
CREATE TABLESPACE CRITTS01 IN XYZDB01
   FREEPAGE 0
   USING STOGROUP SYSDEFLT
   PRIQTY 40 SECQTY 40
   SEGSIZE 8;
COMMIT;
CREATE TABLE XYZDB01.CRITTB01(
   EMPNO     INTEGER  NOT NULL
   ,FNAME     CHAR(20) NOT NULL WITH DEFAULT
   ,LNAME     CHAR(30) NOT NULL WITH DEFAULT
   ,JOBCODE   CHAR(03) NOT NULL WITH DEFAULT
) IN XYZDB01.CRITTS01;
COMMIT;
CREATE UNIQUE INDEX XYZDB01.CRITIX01
   ON XYZDB01.CRITTB01 (EMPNO)
   USING STOGROUP SYSDEFLT PRIQTY 40 SECQTY 40
   CLUSTER;
COMMIT;
```

2 Create DB2 objects.

Using the DDL from Step 1 on page 527, create the table space, table, and indexes. The table space and table must be defined exactly as they were before the drop.
If you create the table space more than once (for example, if it is not created correctly the first time), an extra step is required to accomplish the recovery. You must specify a TORBA/TOLOGPOINT value in the NGT Recover control cards to tell NGT Recover when to stop applying log records. Use DSN1LOGP to identify the log point of the first CREATE table space after the drop. This is the log point you should specify with the TORBA/TOLOGPOINT option. If you do not specify a TORBA/TOLOGPOINT value in this case, the recovery will result in an empty table space.

3 Obtain internal object identifiers for the dropped objects.

To run a recovery using the image copy of the dropped table space, you need the internal object identifiers assigned to the database, table space, and table when they were originally created. Because the table space has been dropped, the IDs for the table space and table are no longer available in the DB2 system catalog. You can use the BMC Log Master for DB2 product to find SYSCOPY entries and IDs.

You can also find the object identifiers by using the IBM DB2 stand-alone utility DSN1PRNT to print a range of pages from the image copy data set. The JCL to run DSN1PRNT is shown in the following figure.

Figure 146: Example JCL to run DSN1PRNT

```
//RUNPRNT EXEC PGM=DSN1PRNT,
//              PARM='FORMAT,FULLCOPY,PRINT(000,003)'
//STEPLIB DD DSN=SYS2.DB2.PROD.DSNLOAD,DISP=SHR
//          DD DSN=SYS3.DBAJ.DSNEXIT,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSUT1 DD DSN=RDAMSM.ICDBAJ.CRITTS01.G0003V00,DISP=SHR
```

Output from this example job is shown in the following figure.

Figure 147: Example DSN1PRNT output with the object identifiers of the dropped table space

<table>
<thead>
<tr>
<th>DSN1999I START OF DSN1PRNT FOR JOB RDAMSMPR RUNPRNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN1998I INPUT DSNAME = RDAMSM.ICDBAJ.CRITTS01.G0003V00 , SEQ</td>
</tr>
<tr>
<td>HEADER PAGE: PGCMB='10'X PGLGRRBA='000000000000'X PGLGID='FF'X PGNUMBER='000000'X</td>
</tr>
<tr>
<td>PGLGSRBA='38'X HPGOBID='014C0002'X HPGHPRF='00000001'X HPGREL='00'X</td>
</tr>
<tr>
<td>HPGRORBA='000000000000'X HPGTSTMP='000100000000000000'X HPGSSNM='DBAJ'</td>
</tr>
<tr>
<td>HPGRFOID='00001'X HPGPGSZ='100000'X HPGSGS='0008'X HPGRPARTN='000000'X</td>
</tr>
<tr>
<td>HPGZPNUM='000000'X HPGZNUM='0000'X HPGBL='0001'X HPGRID='0003'X HPGMAXL='0039'X</td>
</tr>
<tr>
<td>HPGNUM='0004'X HPGBID='000X' HPGCONT='19914121081533515711'X</td>
</tr>
<tr>
<td>HPGSNAM='SYSDEFLT' HPGCATN='DBAJCAT' HPGBRBA='000308285630'X FOEND='E'</td>
</tr>
<tr>
<td>SPACE MAP PAGE: PGCMB='10'X PGLGRRBA='000000000000'X PGLGID='FF'X</td>
</tr>
<tr>
<td>PGNUMBER='00000001'X</td>
</tr>
<tr>
<td>PGLGSRBA='30'X SEGNUM='0152'X SEGFREE='0151'X SEGENT='0002'X SEGSIZE='0008'X</td>
</tr>
<tr>
<td>SEGLEN='00000001'X FOEND='E'</td>
</tr>
<tr>
<td>FIRST PART OF SEGMENTED SPACE MAP:</td>
</tr>
<tr>
<td>SEG 0001 00000000000003CO 300000000</td>
</tr>
<tr>
<td>SECOND PART OF SEGMENTED SPACE MAP:</td>
</tr>
</tbody>
</table>
In the output from DSN1PRNT in the preceding example, examine the following fields to find the relevant object identifiers:

- **HPGOBID**: The first two bytes of this field are the hexadecimal representation of the database identifier (DBID) of the database (X’014C’ in the example output displayed in the preceding example). The last two bytes are the hexadecimal representation of the page set identifier (PSID) of the table space (X’0002’ in the example output displayed in the preceding example).

- **PGSOBD**: This field is found in the header information for each row. It is the hexadecimal representation of the object identifier (OBID) of the table for that row (X’0003’ in the example output in the preceding example).

**Note**

In this example, there is only one table. Because it is a segmented table space, the OBID for the table is also found in the space map segment entries. If your table space has several tables, you may have to print several pages to locate all of the IDs so that you can review the lengths of data in order to distinguish between the tables.

4 Obtain internal object identifiers for the newly created objects.

You can use SPUFI to query the DB2 system catalog for the new internal object identifiers. See the following figure for an example.
You can also accept the default values for the target DBID and PSID, as well as for the OBID if you are working with a single-table table space.

**Figure 148: Example SPUFI to locate new internal object identifiers**

```sql
SELECT DBID, PSID
FROM SYSIBM.SYSTABLESPACE
WHERE DBNAME = 'XYZDB01'
AND NAME   = 'CRITTS01';
```

<table>
<thead>
<tr>
<th>DBID</th>
<th>PSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>332</td>
<td>12</td>
</tr>
</tbody>
</table>

DSNE610I NUMBER OF ROWS DISPLAYED IS 1

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

```sql
SELECT NAME, OBID
FROM SYSIBM.SYSTABLES
WHERE DBNAME = 'XYZDB01'
AND TSNAME = 'CRITTS01';
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRITTB01</td>
<td>13</td>
</tr>
</tbody>
</table>

DSNE610I NUMBER OF ROWS DISPLAYED IS 1

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

5 Create NGT Recover JCL.

Use the DROPRECOVERY and OBIDXLAT options with NGT Recover to accomplish this recovery. Use the OBIDs from the old image copy as source OBIDs and the OBIDs from the newly created object as target OBIDs. Specify the log point of the image copy as the starting point for NGT Recover to apply log records. If you do not know the log point of the image copy, use the value of HPGRBRBA in the output from DSN1PRNT. (For an example, see Figure 147 on page 528.)

**Note**

If the table space was not updated for a long time after the copy was made, the log point value might be too low. A low value is harmless but causes more log to be read. If this is the case, you can compare values in the output of PRINT LOG MAP to obtain an approximate log point. Ensure that the log point you use is not too high.

The NGT Recover JCL and control statements used for this example are shown in the following figure.

**Figure 149: Example drop recovery NGT Recover JCL**

```sql
RECOVER EXEC PGM=AFRMAIN,
```

//RECOVER THE DROPPED TABLESPACE
//USING IMAGE COPY INPUT AND LOG RECORDS

BMC Next Generation Technology Recover for DB2 for z/OS Reference Manual
// REGION=OM,
// PARM="DBAJ.CRITRECOV.NEW,MSGLEVEL(1)"
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(2,1))
//SYSIN DD *
RECOVER TABLESPACE XYZDB01.CRITTS01
   DROPRECOVERY
   INCOPY
   FULL  DSNAME
      RDSM.ICDBAJ.CRITTS01.G0003V00
      LOGPOINT X'0003082F630'
      OBIDXLAT DBID(X'014C',X'014C')
      PSID(X'0002',X'000C')
      OBID(X'0003',X'000D')
REBUILD INDEX(ALL) TABLESPACE XYZDB01.CRITTS01
/*

6 Run the recovery.

The job should end with a condition code 4 and a warning indicating that an
image copy is required for the table space. If you would like to create a copy at
recovery time, you can add OUTCOPY YES to the recovery and avoid the
condition code 4.

Example 2: Recovering a dropped table in a
segmented table space containing multiple
tables

One of your production table spaces contains multiple tables. Approximately a week
ago, one table was dropped from the table space. Since then, the other tables in the
table space have been updated.

Your challenge is how to recover this dropped table without regressing the updates
that occurred to the other tables.

Using the NGT Recover OBIDXLAT and INDEPENDENT OUTSPACE options, your
problem is solved. To preserve the updates to the other tables, this process recovers
the entire original table space into a temporary table space at the point just prior to
dropping the table. The rows from the temporary table space table that corresponds
to the dropped table will then be inserted into the newly created table within the
original table space.

To recover a dropped table in a segmented table space containing multiple
tables

1 Obtain data definitions.
Obtain the definitions for the table space and all of its tables. If you also want to create the indexes, obtain the definitions of the indexes. (Though it is not necessary to include the indexes in this process, you might want to create them so that you can more easily execute verification queries later.)

**Note**

You can obtain the object definitions by using a tool such as one of the following BMC products: CATALOG MANAGER for DB2, ALTER for DB2, or CHANGE MANAGER for DB2.

2 Create DB2 objects.

Using the DDL you obtained in Step 1 on page 531, create the temporary table space with table and table column definitions identical to the original table space. (Do not change any characteristics of the table space that affect its internal structure.) Also, create the dropped table, along with its indexes, within the original table space.

3 Obtain internal object identifiers for the original table space.

Obtain the DBID and PSID of the original table space. Obtain the OBIDs of each of the tables within the table space, including the OBID of the table that was dropped. To obtain this information, print the first two pages of the image copy by using the IBM DB2 stand-alone utility DSN1PRNT. For an example of DSN1PRNT JCL, see the following figure. Figure 151 on page 532 shows an example of DSN1PRNT output.

The DBID, PSID, and OBIDs (except those for the original dropped table) are also available from the DB2 system catalog. See Step 4 on page 529 for more information about obtaining object identifiers from the catalog.

**Figure 150: Example JCL to run DSN1PRNT to obtain object identifiers for original table space**

```plaintext
//DSN01 EXEC PGM=DSN1PRNT,PARM='PRINT(0,1),FORMAT'
//STEPLIB      DD DSN=SYS2.DB2.PROD.DSNLOAD,DISP=SHR
//             DD DSN=SYS3.DB2R.DSNEXIT,DISP=SHR
//SYSPRINT     DD SYSOUT=*  
//SYSUT1       DD DISP=SHR,DSN=RDADMB.COPY8.G0028V00    <= IMAGE COPY
```

**Figure 151: Example DSN1PRNT utility output with object identifiers for original table space**

| Example 2: Recovering a dropped table in a segmented table space containing multiple tables |

<table>
<thead>
<tr>
<th>IDSN19991 START OF DSN1PRNT FOR JOB RDAMSMPR RUNPRNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN19981 INPUT DSNAME = RDAMSM.CRITTS01.G003V00 . SEQ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HEADER PAGE: PGCMB='10'X PLOGRBA='000000000000'X PLOGID='FF'X PNUMBER='000000'X</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGFLAGS='38'X HPGBID='014D0002'X HPGPREF='00000001'X HPGREL='C6'X</td>
</tr>
<tr>
<td>HPGTORBA='000000000000'X HPGTSTMP='00010101000000000000'X HPGSSNM='DBAJ'</td>
</tr>
<tr>
<td>HPGF0ID='0001'X HPGPSGSZ='1000'X HPGSSGZ='0008'X HPGPARTN='0000'X</td>
</tr>
<tr>
<td>HPZPNUM='000000'X HPZNUMP='0000'X HPZTBL='0001'X HPRDIR='0003'X HPGMAXL='0039'X</td>
</tr>
<tr>
<td>HPNUMCO='0004'X HPFFLAGS='0001'X HPGCONTM='1994121208153351711'X</td>
</tr>
<tr>
<td>HPSSGAM='SYSDEFALT' HPGVCATN='DBAJCAT' HPGBPBB='000308256030'X FOEND='N'</td>
</tr>
<tr>
<td>SPACE MAP PAGE: PGCMB='00'X PLOGRBA='000000000000'X PLOGID='FF'X PNUMBER='000000'X</td>
</tr>
</tbody>
</table>
The first two bytes of HPGOBID are the hexadecimal representation of the DBID (X'014D' in the example output in the preceding example). The last two bytes of HPGOBID are the hexadecimal representation of the PSID of the table space (X'000A' in the example output in the preceding example).

Notice that the OBIDs X'000B', X'000E' and X'0011' are found in the segment entries. By comparing these to the OBIDs in SYSIBM.SYSTABLES for this table space, you can determine which tables are still present and which table has been dropped.

4 Obtain internal object identifiers for the temporary table space.

Obtain the DBID and PSID values of the temporary table space. Obtain the OBID values of each of the tables within this table space. You can query the DB2 system catalog to obtain these identifiers. For an example query, see the following figure.

Figure 152: Example query for the DBID, PSID, and OBID values for the temporary table space
### Example 2: Recovering a dropped table in a segmented table space containing multiple tables

5. Find the log record sequence number (LRSN) prior to dropping the table.

   You can use the DSN1LOGP utility to find the LRSN at which the table was dropped. Because the table was dropped after the last copy, you use the LRSN of this copy for the DSN1LOGP STARTRBA or STARTLRSN. The log output from the DSN1LOGP utility is then searched to find the update to the SYSDATABASE table space, DBID(0006) and OBID(0009), which removed the definition for the dropped table. The URID of this event (or for data sharing, the LRSN of the log record at the URID RBA) is used for the TORBA/TOLOGPOINT value in the recovery. Refer to Figure 154 on page 536 and Figure 155 on page 536 for sample JCL and output.

   You could also use Log Master to find the LRSN at which the table was dropped.

6. Create and run NGT Recover JCL.

   The original table space is recovered into the temporary table space using the NGT Recover INDEPENDENT OUTSPACE option. The OBIDXLAT feature translates the internal identifiers found in the image copy to the identifiers of the temporary table space. Because the table space was not dropped, the SYSCOPY and SYSLGRNX information is still available and the DROPRECOVERY option is not required. For an example of NGT Recover drop recovery JCL, see the following figure.

#### Figure 153: Example NGT Recover drop recovery JCL

```sql
SELECT A.DBNAME, A.TSNAME, A.NAME,
       HEX(A.DBID), HEX(B.PSID), HEX(A.OBID)
FROM SYSIBM.SYSTABLES A,
     SYSIBM.SYSTABLESPACE B
WHERE A.TSNAME = B.NAME
  AND A.DBNAME = B.DBNAME
  AND A.TSNAME = 'DMBSPAC2'
  AND A.DBNAME = 'DMBDROPS'
ORDER BY A.NAME;
```

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>NAME</th>
<th>DBID</th>
<th>OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMBDROPS</td>
<td>DMBSPAC2</td>
<td>DMBTBL1</td>
<td>015E</td>
<td>0002</td>
</tr>
<tr>
<td>DMBDROPS</td>
<td>DMBSPAC2</td>
<td>DMBTBL2</td>
<td>015E</td>
<td>0002</td>
</tr>
<tr>
<td>DMBDROPS</td>
<td>DMBSPAC2</td>
<td>DMBTBL3</td>
<td>015E</td>
<td>0002</td>
</tr>
</tbody>
</table>
Example 3: Recovering a dropped table from a simple table space

The table from a simple table space has been accidentally dropped. The table space contains only one table. You are required to recover the table space just prior to when the drop occurred.

To recover a dropped table from a simple table space

1 Obtain data definitions.

Obtain the definition for the dropped table. You cannot change any of the table’s characteristics that affect the internal structure. If you also want to create the indexes, obtain the definitions of the indexes.

Note

You can obtain the object definitions by using a tool such as one of the following BMC products: CATALOG MANAGER for DB2, ALTER for DB2, or CHANGE MANAGER for DB2.

2 Create DB2 objects.

Using the DDL obtained in Step 1 on page 535, create the table and indexes.

3 Find the LRSN prior to dropping the table.

You can use the DSN1LOGP utility to find the LRSN at which the table was dropped. Because the table was dropped after the last copy, you supply this LRSN for the DSN1LOGP STARTRBA. You could also use Log Master to find the LRSN.

7 Insert rows from the temporary table into the newly created table.

Use SQL to insert the rows from the temporary table into the original table.

Example 3: Recovering a dropped table from the temporary table into the newly created table.

INSERT INTO ORIGINAL.DMBTBL2
SELECT * FROM TEMP.DMBTBL2;
Search the log output from the DSN1LOGP utility to find the update to the SYSDBASE table space, DBID(0006) and OBID(0009), which removed the definition for the dropped table. Use the URID of this event (or for data sharing, the LRSN of the log record at the URID RBA) for the TORBA/TOLOGPOINT value in the recovery. Locate the time frame of an archive log that would contain the DROP TABLE statement for coding in this example. (DSNJU003 produces a print log map that might help you locate this information.)

The following figure shows example JCL for the DSN1LOGP utility.

**Figure 154: Example DSN1LOGP utility JCL**

```plaintext
//DSN01 EXEC PGM=DSN1LOGP
//STEPLIB DD DSN=SYS2.DB2.PROD.DSNLOAD,DISP=SHR
//                  DD DSN=SYS2.DB2.PROD.DSNSAMP,DISP=SHR
//                  DD DSN=SYS3.DBAJ.DSNEXIT,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSSUMRY DD SYSOUT=*  
//ARCHIVE DD DSN=DBAJCAT.ARCHLOG1.A0000567,DISP=SHR
//SYSIN    DD *  
STARTRBA(00039F08E8D2) DBID(0006) OBID(0009)  
```

The following figure shows example DSN1LOGP utility output.

**Figure 155: Example DSN1LOGP utility output**

```
-- SEARCH CRITERIA -----------------------------------------------
STARTRBA(00039F08E8D2) ENDRBA(FFFFFFFFFFFFF)
SUMMARY(NO)
DATAONLY(NO)
SYSCOPY(NO)
ALL URIDS ==== YOU MAY SPECIFY URID(XXXXXXXXXXXX)
ALL LUWIDS ==== YOU MAY SPECIFY LUWID(NNNNNNNN.LLLLLLL.XXXXXXXXXX.XXXX)
DBID(0006) OBID(0009)
ALL PAGES ==== YOU MAY SPECIFY PAGE(XXXXXX) MANY TIMES
ALL TYPES ==== YOU MAY SPECIFY TYPE(XX)
ALL SUBTYPES ==== YOU MAY SPECIFY SUBTYPE(XX)
33===================================================================

DSN1212I DSN1LGRD FIRST LOG RBA ENCOUNTERED 00039F08E8D2
(LINES OMITTED)
00039F0A80E URID(00039F0A578A) LRSN(AE68AC13F86E) DBID(0006) OBID(0009) PAGE(000243) TYPE( UNDO  REDO ) SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE) CLR(NO)  
PROCNNAME(DSNILGW)
*LRH* 0049A097D 06000001 0E800000 C386CDD9 0000C386 D41E0126 0000C386 D41EA6E8 * F ' Cf R CfM CmFm AC13F68E 0000  
*LG** 10000600 09000243 27000000 C386D115 0000  
* CfJ 0000 0012000C 00170900 00160000 00000002 4312  
00039F0AA959 URID(00039F0A578A) LRSN(AE68AC13F871) DBID(0006) OBID(0009) PAGE(000243) TYPE( UNDO  REDO ) SUBTYPE(DELETE IN A DATA PAGE) CLR(NO)  
PROCNNAME(DSNIDILS)
*LRH* 00D5004A 06000001 0E800000 C386CDD9 0000C386 D49B0126 0000C386 D49BAE68 * N  
```
4 Create and run NGT Recover JCL.

Recover the table space using the OBIDXLAT option to translate the OBID found in the image copy to the OBID of the newly created table. Because the table space was not dropped, the SYSCOPY and SYSLOGRX information is still available and the DROPRECOVERY option is not required.

NGT Recover generates an image copy to allow for recoverability after the translation. The NGT Recover OUTPUT command causes dynamic allocation of the image copy. (See “OUTPUT command” on page 142 for details regarding the OUTPUT command and dynamic allocation of output data sets.) The following figure provides example NGT Recover JCL for this step.

**Figure 156: Example NGT Recover JCL with dynamically allocated output copy**

```
//RECOVER EXEC PGM=AFRMAIN,REGION=5M,
//              PARM='DBAJ,DMBBASIC,NEW,MSGLEVEL(1)'
//STEP11B DD DISP=SHR,DSN=product.libraries
//         DD DISP=SHR,DSN=DB2.DSNEXIT
//         DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
//OUTPUT CP00001
   DSNNAME RDADMB.COPY8(+1)
   UNIT SYSDA
   CATLG YES
   MODELDCB SYS1.MODEL
   SPACE (20,1) CYL
   RECOVER TABLESPACE DMBDROPR.DMBSPAC1
   OUTCOPY YES
   OUTCOPYDDN CP00001
   OBIDXLAT OBID( X'3', X'4')
   TORBA X'000343F5F6EC'
```
Example 4: Recovering a dropped table space using a pack backup

A table space at your company has been accidentally dropped. The most recent image copy of this table space is more than three weeks old. However, you know that your DASD management group makes pack backups of all DB2 data sets every night while DB2 is down.

You have been assigned to recover this table space to the point in time at which it was dropped. To avoid processing three weeks of log data, you decide to use the latest pack dump as input.

To recover a dropped table space using a pack backup

1 Obtain data definitions.

Using a tool such as the BMC CATALOG MANAGER for DB2, ALTER for DB2, or CHANGE MANAGER for DB2 product, obtain the data definitions of the table space and table you want to recover. Also, obtain definitions for the indexes because they were also dropped. (If you use CHANGE MANAGER for DB2, it can systematically change object names using a name mask in a migrate profile, which could be helpful if many table spaces are involved.) You cannot change any of the characteristics of the table space or table that affect their internal structure. The DDL for this example is shown in the following figure.

Figure 157: Example DDL for the dropped table space

```sql
CREATE TABLESPACE CRITTS01 IN ABCDB01
   FREEPAGE 0
   USING STOGROUP SYSDEFLT
   PRIQTY 40 SECQTY 40
   SEGSIZE 8;
COMMIT;
CREATE TABLE ABCDB01.CRITTB01
   (EMPNO     INTEGER  NOT NULL
    ,FNAME     CHAR(20) NOT NULL WITH DEFAULT
    ,LNAME     CHAR(30) NOT NULL WITH DEFAULT 3
    ,JOBCODE   CHAR(03) NOT NULL WITH DEFAULT 3
   )
   IN ABCDB01.CRITTS01;
COMMIT;
CREATE UNIQUE INDEX ABCDB01.CRITIX01
   ON ABCDB01.CRITTB01 (EMPNO)
   USING STOGROUP SYSDEFLT PRIQTY 40 SECQTY 40
   CLUSTER;
COMMIT;
```

2 Create DB2 objects.

Using the DDL from Step 1 on page 538, create the table space, table, and indexes. The table space and table must be defined exactly as they were before they were dropped.
3 Restore the backup.

Using whatever backup and restore mechanism is in place, restore the table space and index data sets.

The example in the following figure uses the IBM utility DFDSS to restore the data sets.

**Figure 158: Example DFDSS JCL to restore the table space data set**

```plaintext
//ADRDSSU EXEC PGM=ADRDSSU,REGION=0M
//SYSPRINT DD SYSOUT=*.
//TAPE DD DSN=RDAMSM.ABCDB01.DUMP, DISP=SHR
//SYSIN DD *
//REST DD
//DS( 
//  INCLUDE(DBAJCAT.DSNDBC.ABCDB01.**) )
//IDD(TAPE) -
//REPLACE -
//ODY(DEV165) -
//CATALOG
/*
```

4 Create NGT Recover JCL.

You need to use the DROPRECOVERY and OBIDXLAT options to accomplish this recovery.

Using the LOGONLY option indicates that there is no image copy input and log records should be applied beginning after the **HPGRBRBA** value in the header page of the table space data set. The following figure shows the example NGT Recover JCL.

**Figure 159: Example NGT Recover JCL for recovery using a pack backup**

```plaintext
//RECOVER EXEC PGM=AFRMAIN,REGION=5M,
//  PARM='DBAJ,DMBBASIC,NEW,MSGLEVEL(1)' 
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSIN DD *
//RECOVER TABLESPACE ABCDB01.CRITTS01
//DROPRECOVERY
//OBIDXLAT DBID (X’014F’,X’014F’)
```
5 Run the recovery.

The job should end with a condition code 4 and a warning message indicating that an image copy is required for this table space. If you want to create a copy during the recovery and avoid the condition code 4, you can add OUTCOPY YES to the recovery.
Migrating data

This chapter provides examples for data migration using the NGT Recover INCOPY and OBIDXLAT keywords (no Recovery Management solution password required).

Overview of migrating data

NGT Recover supports data migration from one DB2 table space to another. Table spaces may reside on the same DB2 subsystem or on different DB2 subsystems or data sharing groups.

Note

If you own a Recovery solution, the migration process is simplified with the Copy Migration feature. Refer to the Recovery Management for DB2 User Guide for information on how to use the IMPORT (“IMPORT” on page 290) and MIGRATE (“MIGRATE” on page 270) commands. These commands eliminate the need to code the INCOPY and OBIDXLAT keywords.

Also, note that Online Consistent Copy can create a consistent copy without a quiesce by using Instant Snapshots. For examples, see the Recovery Management for DB2 User Guide.

Finally, note that there are special considerations for migrating versioned tables. For more information, see “Handling DB2 versioning information” on page 513.
The following figure shows how you can use NGT Recover to migrate data from a production table space to a table space on a different subsystem designated for high-level query functions.

Figure 160: NGT Recover migration within the same complex

In this example, it is assumed that both DB2 subsystems run on systems that share a common DASD pool.

You use NGT Recover to build a table space and index set for the receiving DB2 system by using copies, change accumulation files, and logs made for recovery purposes on the sending DB2 system. To perform the migration, you use the INDEPENDENT OUTSPACE option and the OBDIXLAT option.

You can migrate the indexes using copies and logs in the same manner as the table space. Alternatively, you can rebuild the indexes by specifying the newly migrated table space with the INDEPENDENT INTO TBLSPACE option. In either case, the structure of the indexes cannot be different in this scenario. (Some indexes may be eliminated, however.)

The following figure demonstrates how you can use NGT Recover to migrate data to remote sites. In this example, the recovery resources on the sending subsystem are
used to build a reference migration image. A reference migration image is identical to a SHRLEVEL REFERENCE image copy.

**Figure 161: NGT Recover migration to a different complex**

One copy of this image copy may be registered at the sending site for use in normal DB2 recovery operations. Another copy could be sent off site to build a query version of the production table space at a remote location.

At the receiving site, you can use the INCOPY option of NGT Recover to restore the reference migration image to the query version of the object.

Indexes can also be migrated using the reference migration image. However, if you rebuild the indexes instead, the index structure may be different on the query system to optimize query access (for example, using QMF).

### Defining target objects for migration

In order to successfully migrate data from a source image copy to a target space, the source and target object definitions must be identical.

You cannot change any of the characteristics of the table space or tables that affect their internal structure. Target index definitions can differ if the target indexes are to be rebuilt rather than recovered from a source image copy.
When you are performing a migration, the source definition must match the target definition, including the number of partitions. For the migration of a partition-by-growth (PBG) universal spaces where the number of source data sets differs from the number of target data sets, NGT Recover determines the number of source data sets and acts accordingly.

You can obtain the object definitions by using a tool such as one of the following BMC products: CATALOG MANAGER for DB2, ALTER for DB2, or CHANGE MANAGER for DB2. (If you use CHANGE MANAGER for DB2, it can systematically change object names using a name mask in a migrate profile, which could be helpful if many table spaces are involved).

If you have a Recovery Management solution password, CHANGE MANAGER supports the EXPORT and IMPORT commands (not MIGRATE) of the Copy Migration feature. For more information, see the CHANGE MANAGER user documentation and the Recovery Management for DB2 User Guide.

It is important to be aware of any alterations to source definitions. For example, assume Source_Table_A has been defined with 5 fixed-length columns. Subsequently, Source_Table_A is altered to add a variable-length column and the source image copy is created before a REORG is executed. Defining Target_Table_A with 6 columns (5 fixed-length and 1 variable-length) and using the source image copy for migration will result in a data incompatibility error (00C90101) when trying to access the target data. In this case, Target_Table_A needs to be defined with 5 fixed-length columns, and then altered to add the variable-length column. (If you use CHANGE MANAGER for DB2, it will identify these types of changes to source objects and build target definition and actions appropriately).

Example 1: Creating a consistent database for query and review

Every evening at midnight, a version of an order and shipping database must be migrated to a query database. To create management reports, you must be able to execute long-running queries on the query database without interfering with online transactions.

Orders and shipping data are added by the online transactions. These transactions cannot interfere with the consistency of a series of queries against the query database that remains static while you create the management reports.
In this example, the databases are defined on the same DB2 subsystem. Your challenge is to create a consistent database for query and review (see the following figure).

**Figure 162: Creating a consistent database for query and review**

If you have the BMC CHANGE MANAGER for DB2 product, you can use it to automate this entire process. CHANGE MANAGER generates the JCL and the syntax.

**To create a consistent database for query and review**

1. Obtain data definitions.

   Obtain the definitions of the database, table spaces, and tables from the online application. These are the source objects. Also, obtain definitions for the indexes if you want to create them.

   **Note**

   See “Defining target objects for migration” on page 543 for more information.

   The DDL for this example is shown in the following figure.

   **Figure 163: Data definitions for source objects**

```
CREATE DATABASE AFRORDER;
CREATE TABLESPACE AFRORDER IN AFRORDER USING STOGROUP PROD1;
CREATE TABLESPACE AFRSHIPS IN AFRORDER USING STOGROUP PROD1;
CREATE TABLE AFRORDER
    (ORDERNUM INT
    ITEMCODE INT
    PRICE   DECIMAL (5,2)
    QUANTITY SMALLINT
    TOTAL    DECIMAL (7,2)
    ) IN AFRORDER.AFRORDER;
CREATE TABLE AFRSHIPS
```
2 Create the database.

Modify the DDL for the source objects to change the names so that table spaces and tables are distinctly defined. You might want to create a new database as shown in the following figure. Also, consider other possible changes such as the STOGROUP and index type. The following figure shows the DDL for the target objects.

Figure 164: Data definitions for target objects

```
CREATE DATABASE AFRSUMOR;
CREATE TABLESPACE AFRSUMOR IN AFRSUMOR USING STOGROUP MIS1;
CREATE TABLESPACE AFRSUMSH IN AFRSUMOR USING STOGROUP MIS1;
CREATE TABLE AFRSUMORDER ( ORDERNUM INT
ITEMCODE INT
PRICE DECIMAL (5,2)
QUANTITY SMALLINT
TOTAL DECIMAL (7,2))
IN AFRSUMOR.AFRSUMOR;
CREATE TABLE AFRSUMSHIPS ( ORDERNUM INT
DATEORDERED DATE
DATESHIPPED DATE )
IN AFRSUMOR.AFRSUMSH;
CREATE UNIQUE INDEX AFRORDSI ON AFRSUMORDER (ORDERNUM)
USING STOGROUP MIS1;
CREATE UNIQUE INDEX AFRSHPSI ON AFRSUMSHIPS (ORDERNUM)
USING STOGROUP MIS1;
```

3 Obtain internal IDs.

You need the internal DB2 OBIDs for the source and target objects to set up the migration. You can use SQL to obtain these identifiers from the DB2 system catalog. See the following figure for an example.

**Note**

You can accept the default values for the source DBID and PSID. If the table space contains only one table, you can also accept the default values for the source OBIDs. NGT Recover extracts this information from the catalog. These IDs are in decimal. Because a new database is created, all of the IDs except for the DBID are the same in this example. They are included in the migration job for documentation purposes.

Figure 165: Example SQL and output to select source and target objects

```
SELECT NAME, DBID, PSID FROM SYSIBM.SYSTABLESPACE
WHERE NAME = 'AFRORDER' OR NAME = 'AFRSUMOR' OR NAME = 'AFRSHIPS' OR NAME = 'AFRSUMSH';
```
<table>
<thead>
<tr>
<th>NAME</th>
<th>DBID</th>
<th>PSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFRORDER</td>
<td>352</td>
<td>2</td>
</tr>
<tr>
<td>AFRSHIPS</td>
<td>352</td>
<td>7</td>
</tr>
<tr>
<td>AFRSUMOR</td>
<td>449</td>
<td>2</td>
</tr>
<tr>
<td>AFRSUMSH</td>
<td>449</td>
<td>7</td>
</tr>
</tbody>
</table>

SELECT CREATOR, NAME, INDEXTYPE, DBID, ISOBID FROM SYSIBM.SYSINDEXES WHERE NAME = 'AFRORDI' OR NAME = 'AFRORDSI' OR NAME = 'AFRSHPI' OR NAME = 'AFRSHPSI';

<table>
<thead>
<tr>
<th>CREATOR</th>
<th>NAME</th>
<th>INDEXTYPE</th>
<th>DBID</th>
<th>ISOBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNERID</td>
<td>AFRORDI</td>
<td>2</td>
<td>352</td>
<td>5</td>
</tr>
<tr>
<td>OWNERID</td>
<td>AFRORDSI</td>
<td>2</td>
<td>352</td>
<td>5</td>
</tr>
<tr>
<td>OWNERID</td>
<td>AFRSHPI</td>
<td>2</td>
<td>449</td>
<td>10</td>
</tr>
<tr>
<td>OWNERID</td>
<td>AFRSHPSI</td>
<td>2</td>
<td>449</td>
<td>10</td>
</tr>
</tbody>
</table>

SELECT NAME, OBID FROM SYSIBM.SYSTABLES WHERE DBNAME = 'AFRORDER';

<table>
<thead>
<tr>
<th>NAME</th>
<th>OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFRORDER</td>
<td>3</td>
</tr>
<tr>
<td>AFRSHIPS</td>
<td>8</td>
</tr>
</tbody>
</table>

SELECT NAME, OBID FROM SYSIBM.SYSTABLES WHERE DBNAME = 'AFRSUMOR';

<table>
<thead>
<tr>
<th>NAME</th>
<th>OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFRSUMORDER</td>
<td>3</td>
</tr>
<tr>
<td>AFRSUMSHIPS</td>
<td>8</td>
</tr>
</tbody>
</table>

4 Quiesce the source objects.

Issue a QUIESCE command on the source table spaces. Example JCL and output are shown in the following figures.

**Figure 166: Example QUIESCE JCL**

```
//QUIESCE EXEC DSNUPROC,SYSTEM=DBAN,UID='AFRRESM1',
// UTPROC='',REGION=4M,
// LIB='SYS2.DB2.PROD.DSNLOAD'
//DSNUPROC.SYSPRINT DD SYSOUT=* 
//DSNUPROC.SYSIN DD *
QUIESCE TABLESPACE AFRORDER.AFRORDER
QUIESCE TABLESPACE AFRORDER.AFRSHIPS
```

**Figure 167: Example QUIESCE output**

```
DSNU0001  DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = LSBRESM1
DSNU0501  DSNUGUTC - QUIESCE TABLESPACE AFRORDER.AFRORDER TABLESPACE AFRORDER.AFRSHIPS
DSNU4771  *DSNP DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE AFRORDER.AFRORDER
DSNU4771  *DSNP DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE AFRORDER.AFRSHIPS
```
This quiesce will be the end point (TORBA/TOLOGPOINT) of the NGT Recover request. Note the LRSN.

At this point, the source objects can begin to be updated by other transactions.

If image copies (even SHRLEVEL CHANGE) can be made just before this quiesce point, less information from the log will be needed to accomplish the migration. As a result, the migration will be completed more quickly.

5 Stop the target objects.

Issue a STOP command against the database or table space where the query image will be built.

**Example**

-STOP DATABASE (AFRSUMOR)

6 Run NGT Recover to create the target image.

In this step, you migrate the image to the other spaces using the NGT Recover INDEPENDENT OUTSPACE and OBIDXLAT options.

The data sets that are to receive the migrated data were created when the data definition was done for the target table spaces and index spaces. These data sets are specified in the syntax with their VSAM cluster names.

In the example shown in the following figure, the IDs from Step 3 are used to define the objects required. Note that TORBA (or TOLOGPOINT) LASTQUIESCE is used. If other quiesce points might be created by other activities, use TORBA (or TOLOGPOINT) X’0003D5B9A3C2’—the LRSN from the QUIESCE operation.

**Note**

This step does not affect the source table spaces or indexes.

You can use the RECOVER INDEX command to recover the indexes from image copies and logs if copies of the indexes have been made with BMC Next Generation Technology Copy for DB2 for z/OS, DSN1COPY, or the IBM COPY utility.

**Figure 168: Example JCL to create the target image**

```plaintext
//RECINDEP EXEC PGM=AFRMAIN,REGION=0M,
// PARM='DBAN,AFRAQDN,NEW,MSGLEVEL(1)';
```
//STEPLIB  DD DISP=SHR,DSN=product.libraries 
//         DD DISP=SHR,DSN=DB2.DSNEXIT 
//         DD DISP=SHR,DSN=DB2.DSNLOAD 
//SORTOUT DD DUMMY 
//SYSIN    DD * 
OPTIONS INDEXLOG YES 
RECOVER TABLESPACE AFRORDER.AFRORDER 
   OBIDXLAT DBID(352,449) PSID (2,2) OBID (3,3) 
   INDEP OUTSPACE MODEL 
   DBANCAT.DSNDBC.AFRSUMOR.AFRSUMOR.I0001.A001 
   TORBA LASTQUIESCE 
RECOVER INDEX OWNERID.AFRORDI 
   OBIDXLAT DBID(352,449) PSID (5,5) OBID (3,3) 
   INDEP OUTSPACE MODEL 
   DBANCAT.DSNDBC.AFRSUMOR.AFRORDSI.I0001.A001 
   TORBA LASTQUIESCE 
RECOVER INDEX OWNERID.AFRSHPI 
   OBIDXLAT DBID(352,449) PSID (10,10) OBID (8,8) 
   INDEP OUTSPACE MODEL 
   DBANCAT.DSNDBC.AFRSUMOR.AFRSHPSI.I0001.A001 
   TORBA LASTQUIESCE 
RECOVER TABLESPACE AFRORDER.AFRSHIPS 
   OBIDXLAT DBID(352,449) PSID (7,7) OBID (8,8) 
   INDEP OUTSPACE MODEL 
   DBANCAT.DSNDBC.AFRSUMOR.AFRSUMSH.I0001.A001 
   TORBA LASTQUIESCE 

7 Start the target objects.

Issue a START command to start the target objects. They are now ready for queries.

Example

-START DATABASE (AFRSUMOR)

8 Allow queries.

Queries such as the one shown in the following figure (which requires a table space scan) can be completed without interfering with the online processing. Because data will not be changing on this image, the result of the query will be consistent with other queries made until the data is refreshed again.

Figure 169: Query on target system

```
SELECT COUNT(*), SUM(TOTAL) 
FROM AFR.AFRSUMORDER ORD, 
     AFR.AFRSUMSHIPS SHIP 
WHERE ( 
   DAYS(DATESHIPPED) > DAYS(DATEORDERED) + 2 OR 
   DAYS(DATEORDERED) + 2 < DAYS(CURRENT DATE) AND 
   DATESHIPPED IS NULL 
) 
AND ORD.ORDERNUM = SHIP.ORDERNUM;
```
Example 2: Moving an application to another subsystem with index recovery

For use in auditing, you need to build a shadow of an application that runs on a non-data sharing system. The audit application begins at the end of the year.

**Note**
In this example, RECOVER INDEX is used; therefore, the source and target indexes must be the same.

The table spaces and index spaces for the shadow application are on another DB2 subsystem that is on another computer system with no shared DASD.

**To move an application to another subsystem with index recovery**

1. Obtain an RBA for a consistent view of the source table spaces.

   The table spaces for the application are AFRACCT, AFRTRAN, AFRSUMM, and AFRHIST.

   These table spaces are in database AFRFIN. They need to be at a consistent point for the migration. The point at which logging starts after DB2 goes down normally, flushes all buffers, handles all transactions, and comes up again, can be used for a point of consistency. A sample of messages from DB2 logging is shown in the following figure.

**Figure 170: Sample messages from DB2 logging**

```
DSNR004I > RESTART...UR STATUS COUNTS
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN ABORT=0
DSNR005I > RESTART...COUNTS AFTER FORWARD RECOVERY
IN COMMIT=0, INDOUBT=0
DSNR006I > RESTART...COUNTS AFTER BACKWARD RECOVERY

DSNJ099I > LOG RECORDING TO COMMENCE WITH
STARTRBA=0004D176C05C
```
Rather than having to determine the RBA for the last shutdown, in this example the option TORBA LASTSHUTDOWN is used to allow NGT Recover to select the last shutdown RBA. This technique is used because the system is taken down for year-end synchronization. Another technique for establishing a point of consistency is to issue a QUIESCE command for all of the table spaces in one statement. You could also use the point of consistency from an ARCHIVE LOG MODE(QUIESCE) command.

2 Create migration images.

Run NGT Recover to create migration images. The syntax is completely described in “NGT Recover syntax” on page 83.

In the example JCL shown in the following figure, you create migration images of the table spaces at the point of the pause of the DB2 subsystem that you captured in step 1. These data sets are identical in format to an image copy or DSN1COPY of the spaces. However, you do not register these copies because they are going to be shipped to the audit system. Also, because they are going to a different computer system, you do not catalog these copies with the operating system.

If you want to make registered copies, see “Making copies from the log and earlier copies” on page 567. You may combine these operations.

Note
AFRACCT is partitioned, but the OUTCOPY ASCODED option is used to make a copy that is for all of the partitions.

Figure 171: Example JCL to create migration images with index recovery

```sql
//RECOU4C1 EXEC PGM=AFRMAIN,REGION=0M,
// PARM='DBAN,AFRAQDN,NEW,MSGLEVEL(1)' //STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT // DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSPICK DD SYSOUT=* //ACCTCPY DD DSN=AFR.ACCTCPY1,
// UNIT=CART,
// DISP=(NEW,KEEP) //TRANCPY DD DSN=AFR.TRANCPY1,
// UNIT=CART,
// DISP=(NEW,KEEP) //SUMMCPY DD DSN=AFR.SUMMCPY1,
// UNIT=CART,
// DISP=(NEW,KEEP) //HISTCPY DD DSN=AFR.HISTCPY1,
// UNIT=CART,
// DISP=(NEW,KEEP) //ACCICPY DD DSN=AFR.ACCICPY1,
// UNIT=CART,
// DISP=(KEEP) //TRNICPY DD DSN=AFR.TRNICPY1,
// UNIT=CART,
// DISP=(KEEP) //SUMICPY DD DSN=AFR.SUMICPY1,
```

Example 2: Moving an application to another subsystem with index recovery
3 Obtain data definitions.

Obtain the definitions of the database, table spaces, and tables for which you want to create the audit application for querying. These are the source objects. Also, obtain definitions for the indexes if you want to create them in the audit application.

---

**Note**

See “Defining target objects for migration” on page 543 for more information.

---

You cannot change any of the characteristics of the table space, tables, or indexes that affect their internal structure. The AFRTRAN table space must remain partitioned and use the same key. If you want to change or add to the indexes used, you must use the technique shown in “Example 3: Moving an application to another subsystem with index rebuilding” on page 558.

For this example, you would obtain the DDL shown in the following figure.

---

**Figure 172: Data definitions for source objects**

```sql
CREATE DATABASE AFRFIN;
CREATE TABLESPACE AFRACCT IN AFRFIN USING STOGROUP AFRTEST
NUMPARTS 3;
CREATE TABLESPACE AFRTRAN IN AFRFIN USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRSUMM IN AFRFIN USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRHIST IN AFRFIN USING STOGROUP AFRTEST;
CREATE TABLE AFRACCT

   ACCTNO INT,
   ACCTDESC CHAR(30),
   ACCTTYPE CHAR(10),
   BALANCE DECIMAL (10,2)

) IN AFRFIN.AFRACCT;
CREATE TABLE AFRTRAN

```
4 Obtain object IDs for the source objects.

On the source system, obtain the object IDs. You can do this by using SQL statements to query the catalog or by using some other tool. The example SQL statements and the output are shown in the following figure.

**Note**
You can use the default values for the source DBID and PSID. If the table space contains only one table, you can also use the default values for the source OBIDs. NGT Recover extracts this information from the image copy.

**Figure 173: Example SQL and output to obtain object IDs for source objects**

```sql
SELECT NAME, DBID, PSID FROM SYSIBM.SYSTABLESPACE WHERE NAME = 'AFRTRAN' OR NAME = 'AFRACCT' OR NAME = 'AFRHIST' OR NAME = 'AFRSUMM';
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>DBID</th>
<th>PSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFRACCT</td>
<td>321</td>
<td>2</td>
</tr>
<tr>
<td>AFRHIST</td>
<td>321</td>
<td>8</td>
</tr>
<tr>
<td>AFRSUMM</td>
<td>321</td>
<td>6</td>
</tr>
</tbody>
</table>
5 Create objects on the target system.

Create the database, table spaces, tables, and indexes on the target system using the data definitions in the following figure. They need to be consistent in structure with the source system. (NGT Recover provides migration checking when INCOPY is used.)

**Figure 174: Data definitions for target objects**

```sql
CREATE DATABASE AFRAUDIT;
CREATE TABLESPACE AFRACCT IN AFRAUDIT USING STOGROUP AFRTEST NUMPARTS 3;
CREATE TABLESPACE AFRTRAN IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRSUMM IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRHIST IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLE AFRACCT
  (ACCTNO     INT
  ,ACCTDESC   CHAR(30)
  ,ACCTTYPE   CHAR(10)
  ,BALANCE    DECIMAL (10,2))
IN AFRAUDIT.AFRACCT;
CREATE TABLE AFRTRAN
  (ACCTNO     INT
  ,TRANSDATE  DATE
  ,TRANSDESC  CHAR(40)
  ,CLERKID    CHAR(4)
  ,TOTAL      DECIMAL (10,2))
IN AFRAUDIT.AFRTRAN;
CREATE TABLE AFRSUMM
  (ACCTNO     INT
  ,REPCAT     CHAR (40)
  ,SUMMCAT    CHAR(40)
  ,SUMMDATE   DATE
  ,TOTAL      DECIMAL (10,2))
IN AFRAUDIT.AFRSUMM;
```
6 Obtain IDs on the target system.

On the target system, obtain the object IDs needed for translation. You can do this by using SQL statements to query the DB2 system catalog. The example SQL statements and the output are shown in the following figure.

**Note**

You can use the default values for the target DBID and PSID. If the table space contains only one table, you can also use the default values for the target OBIDs. NGT Recover extracts this information from the catalog.

**Figure 175: Example SQL and output to obtain object IDs for target objects**

```
SELECT NAME, DBID, PSID FROM SYSIBM.SYSTABLESPACE WHERE 
NAME = 'AFRTRAN' OR NAME = 'AFRACCT' OR 
NAME = 'AFRHIST' OR NAME = 'AFRSUMM';
```

```
<table>
<thead>
<tr>
<th>NAME</th>
<th>DBID</th>
<th>PSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFRACCT</td>
<td>336</td>
<td>2</td>
</tr>
<tr>
<td>AFRHIST</td>
<td>336</td>
<td>8</td>
</tr>
<tr>
<td>AFRSUMM</td>
<td>336</td>
<td>6</td>
</tr>
<tr>
<td>AFRTRAN</td>
<td>336</td>
<td>4</td>
</tr>
</tbody>
</table>
```

```
SELECT CREATOR, NAME, INDEXTYPE, DBID, ISOBID, OBID FROM 
SYSIBM.SYSINDEXES WHERE 
NAME = 'AFRACCTI' OR NAME = 'AFRTRANI' OR 
NAME = 'AFRSUMMI' OR NAME = 'AFRHISTI';
```

```
<table>
<thead>
<tr>
<th>CREATOR</th>
<th>NAME</th>
<th>INDEXTYPE</th>
<th>DBID</th>
<th>ISOBID</th>
<th>OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNERID</td>
<td>AFRACCTI</td>
<td>2</td>
<td>336</td>
<td>14</td>
<td>13</td>
</tr>
</tbody>
</table>
```
Recover with INCOPY and ID translation.

The tapes from the source system are delivered at the target system. You use the external label information to complete the JCL to create data for the target. The ID translation is built from the data in step 4 and step 6. Output copies are made to allow a normal recovery on the audit system, if required.

*Note*

NGT Recover provides migration checking when the INCOPY option is used, and performs header checking and provides messages (BMC40470, BMC40471, and BMC40472) to indicate incompatible migrations. Some examples of incompatible migrations are:

- Migration of a partitioned table space into a nonpartitioned table space
- Migration of a segmented table space into a table space with a different segment size
- Migration into a table space with a different page size

Example JCL for this process is shown in the following figure.

*Note*

To illustrate that you do not need to specify the DBIDs, OBIDs, and PSIDs for a single table table space with INCOPY, this example JCL does not include them.

### Figure 176: Example JCL to migrate data with index recovery

```jcl
//RECOUTC1 EXEC PGM=AFRMAIN,REGION=0M,
// PARM='DBAJ,AFRAQDN,NEW,MSGLEVEL(1),,RDB2STAT(YES)'
//STEP1 DD DISP=SHR,DSN=product.libraries
//DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSPICK DD DSN=AFR.ACCTCPYA,
// UNIT=SYSDA,SPACE=(CYL,(10,10)),
// DISP=(NEW,CATLG)
//TRANCPY DD DSN=AFR.TRANCPYA,
// UNIT=SYSDA,SPACE=(CYL,(10,10)),
// DISP=(NEW,CATLG)
//SUMMCPY DD DSN=AFR.SUMMCPYA,
// UNIT=SYSDA,SPACE=(CYL,(10,10)),
// DISP=(NEW,CATLG)
//HISTCPY DD DSN=AFR.HISTCPYA,
```
Example 2: Moving an application to another subsystem with index recovery
Example 3: Moving an application to another subsystem with index rebuilding

Your new assignment is to build a shadow of an application that runs on a non-data sharing system for use in auditing. At the target site, some new indexes are created to facilitate the audit. The audit application begins at the end of the year.

The table spaces and index spaces for the shadow application are on another DB2 system that is on another computer system with no shared DASD.

Note
This example uses REBUILD INDEX, which allows you to have different source and target indexes.

You can also use this technique when you want to create a different set of indexes for your target spaces on the same DB2 system or at the same site.

To move an application to another subsystem with index rebuilding

1 Obtain an RBA for a consistent view of the source table spaces.

The table spaces for the application are AFRACCT, AFRTRAN, AFRSUMM, and AFRHIST.

The table spaces are in database AFRFIN. They need to be at a consistent point for the migration. The point at which logging starts after DB2 goes down normally, flushes all buffers, handles all transactions, and comes up again can be used for a
point of consistency. A sample of DB2 logging messages is shown in the following figure.

**Figure 177: Sample messages from DB2 logging**

```
DSNR004I > RESTART...UR STATUS COUNTS
  IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN ABORT=0
DSNR005I > RESTART...COUNTS AFTER FORWARD RECOVERY
  IN COMMIT=0, INDOUBT=0
DSNR006I > RESTART...COUNTS AFTER BACKWARD RECOVERY
  .
DSNJ099I > LOG RECORDING TO COMMENCE WITH
  STARTRBA=0004D176C05C
```

**Note**

Rather than having to determine the RBA for the last shutdown, in this example the option TORBA LASTSHUTDOWN is used to allow NGT Recover to select the last shutdown RBA.

This technique is used because the system is taken down for year-end synchronization. Another technique for establishing a point of consistency is to issue a QUIESCE command for all of the table spaces in one statement. (See “Example 1: Creating a consistent database for query and review” on page 544.) You could also use the point of consistency from an ARCHIVE LOG MODE(QUIESCE) command.

2 Create migration images.

Run NGT Recover to create migration images. The syntax is completely described in “NGT Recover syntax” on page 83.

In the example JCL shown in the following figure, you create migration images of the table spaces at the point of the pause of the DB2 subsystem that you captured in the previous step. These data sets are identical in format to an image copy or DSN1COPY of the spaces. However, you do not register these copies because they are going to be shipped to the audit system. Also, because they are going to a different computer system, you do not catalog these copies with the operating system.

If you want to make registered copies, see “Making copies from the log and earlier copies” on page 567.

**Note**

AFRACCT is partitioned, but the OUTCOPY ASCODED option is used to make a copy of all of the partitions.

**Figure 178: Example JCL to create migration images**

```
//RECOUTC1 EXEC PGM=AFRMAIN,REGION=0M,
// PARM='DBAN,AFRAQDN,NEW,MSGLEVEL(1)';
//STEPLIB DD DISP=SHR,DSN=product.libraries
```
3 Obtain data definitions.

Obtain the definitions of the database, table spaces, and tables for which you want to create the audit application for querying. These are the source objects. Also, obtain definitions for the indexes if you want to create these in the audit application.

**Note**

See “Defining target objects for migration” on page 543 for more information.

You cannot change any of the characteristics of the table space or tables that affect their internal structure. The AFRTRAN table space must remain partitioned and use the same key. You may change or add to the other indexes used because you are rebuilding them on the target DB2.

For this example, you would obtain the DDL shown in the following figure.

**Figure 179: Data definitions for the source objects**

```sql
CREATE DATABASE AFRFIN;
CREATE TABLESPACE AFRACCT IN AFRFIN USING STOGROUP AFRTEST NUMPARTS 3;
CREATE TABLESPACE AFRTRAN IN AFRFIN USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRSUMM IN AFRFIN USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRHIST IN AFRFIN USING STOGROUP AFRTEST;
CREATE TABLE AFRACCT

<table>
<thead>
<tr>
<th>ACCTNO</th>
<th>INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCTDESC</td>
<td>CHAR(30)</td>
</tr>
<tr>
<td>ACCTTYPE</td>
<td>CHAR(10)</td>
</tr>
<tr>
<td>BALANCE</td>
<td>DECIMAL (10,2)</td>
</tr>
</tbody>
</table>

 IN AFRFIN.AFRACCT;
```
4 Obtain object IDs for the source objects.

On the source system, obtain the object IDs. You can do this by using SQL statements to query the catalog or by using some other tool. The example SQL statements and the output are shown in the following figure.

**Note**

You can use the default values for the source DBID and PSID. If the table space contains only one table, you can also use the default values for the target OBIDs. NGT Recover extracts this information from the image copy.

**Figure 180: Example SQL and output to obtain object IDs for source objects**

```
SELECT NAME, DBID, PSID FROM SYSTABLESPACE WHERE NAME = 'AFRTRAN' OR NAME = 'AFRACCT' OR NAME = 'AFRHIST' OR NAME = 'AFRSUMM';
```

```
<table>
<thead>
<tr>
<th>NAME</th>
<th>DBID</th>
<th>PSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFRACCT</td>
<td>321</td>
<td>2</td>
</tr>
<tr>
<td>AFRHIST</td>
<td>321</td>
<td>8</td>
</tr>
</tbody>
</table>
```
5 Create objects on the target system.

Create the database, table spaces, and tables on the target system. They need to be consistent in structure with the source system. (NGT Recover provides migration checking when INCOPY is used.)

The partitioned index needs to be the same, but the other indexes can be deleted or changed. You can also create new indexes. In this example, you add a new index, AFRTRANX, as shown in the following figure.

**Figure 181: Data definitions for target objects**

```sql
CREATE DATABASE AFRAUDIT;
CREATE TABLESPACE AFRHOLD1 IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRHOLD2 IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRACCT IN AFRAUDIT USING STOGROUP AFRTEST NUMPARTS 3;
CREATE TABLESPACE AFRTRAN IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRSUMM IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLESPACE AFRHIST IN AFRAUDIT USING STOGROUP AFRTEST;
CREATE TABLE AFRACCT (
    ACCTNO     INT,
    ACCTDESC   CHAR(30),
    ACCTTYPE   CHAR(10),
    BALANCE    DECIMAL (10,2)
) IN AFRAUDIT.AFRACCT;
CREATE TABLE AFRTRAN (
    ACCTNO     INT,
    TRANSDATE  DATE,
    TRANSDESC  CHAR(40),
    CLERKID    CHAR(4),
    TOTAL      DECIMAL (10,2)
) IN AFRAUDIT.AFRTRAN;
CREATE TABLE AFRSUMM (
    ACCTNO     INT,
    REPCAT     CHAR (40),
    SUMMCAT    CHAR(40),
    SUMMDATE   DATE,
    TOTAL      DECIMAL (10,2)
) IN AFRAUDIT.AFRSUMM;
CREATE TABLE AFRHIST (
    ACCTNO     INT,
    TRANSDATE  DATE,
    TRANSDESC  CHAR(40)
) IN AFRAUDIT.AFRHIST;
```
6 Obtain IDs on the target system and set up translation.

On the target system, obtain the object IDs that you need for translation. You can do this by using SQL statements to query the catalog. The example SQL statements and output are shown in the following figure.

**Note**

You can use the default values for the target DBID and PSID. If the table space contains only one table, you can also use the default values for the target OBIDs. NGT Recover extracts this information from the catalog.

**Figure 182: Example SQL and output to obtain object IDs for target objects**

```sql
SELECT NAME, DBID, PSID FROM SYSIBM.SYSTABLESPACE WHERE NAME = 'AFRTRAN' OR NAME = 'AFRACCT' OR NAME = 'AFRHIST' OR NAME = 'AFRSUMM';
```

```
+----------------------------------+
|   NAME   |   DBID    |   PSID    |
+----------------------------------+
| AFRACCT  |       336 |         6 |
| AFRHIST  |       336 |        12 |
| AFRSUMM  |       336 |        10 |
| AFRTRAN  |       336 |         8 |
+----------------------------------+

SELECT NAME, OBID FROM SYSIBM.SYSTABLES WHERE DBNAME = 'AFRAUDIT';
```

```
+-----------+-------+
| NAME      | OBID  |
+-----------+-------+
| AFRACCT   | 13    |
| AFRTRAN   | 14    |
| AFRSUMM   | 15    |
| AFRHIST   | 16    |
+-----------+-------+
```

7 Recover with INCOPY and ID translation.
The tapes from the source system are delivered at the target system. You use the external label information to complete the JCL to create data for the target with NGT Recover. The ID translation is built from the data in step 4 and step 6. Output copies are made to allow a normal recovery on the audit system, if required.

**Note**

NGT Recover provides migration checking when the INCOPY option is used, and performs header checking and provides messages (BMC40470, BMC40471, and BMC40472) to indicate incompatible migrations. Some examples of incompatible migrations are:

- Migration of a partitioned table space into a nonpartitioned table space
- Migration of a segmented table space into a table space with a different segment size
- Migration into a table space with a different page size

Example JCL for this process is shown in the following figure.

**Figure 183: Example JCL to migrate data with index rebuilding**

```jcl
//RECOUTC1 EXEC PGM=AFRMAIN,REGION=0M,
// PARM='DBAJ,AFRAQDN,NEW,MSGLEVEL(1),,RDB2STAT(YES)'
//STEPLIB   DD DISP=SHR,DSN=product.libraries
//          DD DISP=SHR,DSN=DB2.DSNEXIT
//          DD DISP=SHR,DSN=DB2.DSNLOAD
//SYSPICK   DD DSYSOUT=*
//ACCTCPY   DD DSN=AFR.ACCTCPYA,
// UNIT=SYSDA,SPACE=(CYL,(10,10)),
// DISP=(NEW,CATLG)
//TRANCPY   DD DSN=AFR.TRANCPYA,
// UNIT=SYSDA,SPACE=(CYL,(10,10)),
// DISP=(NEW,CATLG)
//SUMMCPY   DD DSN=AFR.SUMMCPYA,
// UNIT=SYSDA,SPACE=(CYL,(10,10)),
// DISP=(NEW,CATLG)
//HISTCPY   DD DSN=AFR.HISTCPYA,
// UNIT=SYSDA,SPACE=(CYL,(10,10)),
// DISP=(NEW,CATLG)
//SORTOUT   DD DUMMY
//SYSIN     DD *
//OPTIONS OUTCOPY ASCODED
RECOVER TABLESPACE AFRAUDIT.AFRACCT
OBIDXLAT RESET DBID(321,336) PSID(2,6)
OBID(9,13)
INCOPY
  FULL DSNAME AFR.ACCTCPY1
  SHRLLEVEL REFERENCE
  INVOLUME 311778
  INDEVT CART
  OUTCOPY YES OUTCOPYDDN(ACCTCPY)
  TOCOPY LASTCOPY
RECOVER TABLESPACE AFRAUDIT.AFRTRAN
OBIDXLAT RESET DBID(321,336) PSID(4,8)
OBID(10,14)
INCOPY
  FULL DSNAME AFR.TRANCPY1
  SHRLLEVEL REFERENCE
  INVOLUME 311763
  INDEVT CART
```
8 Allow processing on the audit system.

The audit system is now ready for use.
Example 3: Moving an application to another subsystem with index rebuilding
Making copies from the log and earlier copies

This chapter provides examples of using the NGT Recover product to create image copies.

Overview of making copies from the log and earlier copies

You can use NGT Recover to make updated image copies from typical recovery resources such as earlier copies, change accumulation files, and DB2 logs.
The following figure shows NGT Recover using a full copy, multiple incremental copies, a change accumulation file (created by using R+/CHANGE ACCUM for DB2), and DB2 log files to produce an updated full image copy.

**Figure 184: Making new copies from old copies and log records**

During an OUTCOPY ONLY operation, NGT Recover writes the output to a sequential image copy data set instead of a DB2 space. This process allows you to make copies without accessing the DB2 space or interfering with normal DB2 access in any way. If you elect to end the process at any of the following points, NGT Recover registers the new copy, if it is registered, as a SHRLEVEL REFERENCE full image copy:

- The log point of a SHRLEVEL REFERENCE incremental copy
- The log point of a quiesce point
- The log point of the last -ARCHIVE MODE(QUIESCE) command
- The last successful subsystem shutdown (in a non-data sharing environment)

In all other cases, NGT Recover registers the new copy as a SHRLEVEL CHANGE copy.

---

**Example 1: Creating a consistent set of copies at a point in time**

You suspect a table space or index has been damaged by an application or a media failure.
If you have a point of consistency but no copy close to the point of the damage, you can use this procedure to create a copy while the investigation of the damage is still underway. If the spaces must be recovered, you will have a single copy created for each space that allows a recovery without requiring log or incremental copies.

**To create a consistent set of copies at a point in time**

1. Find or create a point of consistency.

The most common way to achieve a point of consistency is to issue a QUIESCE command on a set of table spaces in a single statement as shown in the following figure.

**Figure 185: Example JCL to QUIESCE a set of table spaces**

```plaintext
//QUIESCE1 EXEC DSNUPROC,SYSTEM=DBAJ,UID='AFRRM31',
// UTPROC='',REGION=4M,
// LIB='SYS2.DB2.PROD.DSNLOAD'
//DSNUPROC.SYSPRINT DD SYSOUT=*
//DSNUPROC.SYSDISC  DD DUMMY
//DSNUPROC.UPRINT  DD SYSOUT=* 
//DSNUPROC.SYSERR   DD UNIT=WORK,SPACE=(CYL,(2,2))
//DSNUPROC.SYSMAP   DD UNIT=WORK,SPACE=(CYL,(2,2))
//DSNUPROC.SYSIN DD *
QUIESCE TABLESPACE AFRFIN.AFRACCT
TABLESPACE AFRFIN.AFRTRAN
TABLESPACE AFRFIN.AFRSUMM
TABLESPACE AFRFIN.AFRHIST
```

Output from the QUIESCE follows in the following figure.

**Figure 186: Example QUIESCE output**

```plaintext
DSNU050I   DSNUGUTC - QUIESCE TABLESPACE AFRFIN.AFRACCT TABLESPACE
AFRFIN.AFRTRAN TABLESPACE AFRFIN.AFRSUMM TABLESPACE AFRFIN.AFRHIST
DSNU477I > DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE AFRFIN.AFRACCT
DSNU477I > DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE AFRFIN.AFRTRAN
DSNU477I > DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE AFRFIN.AFRSUMM
DSNU477I > DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACE AFRFIN.AFRHIST
DSNU474I > DSNUQUIA - QUIESCE AT RBA 0004D176C05C
DSNU475I   DSNUQUIB - QUIESCE UTILITY COMPLETE, ELAPSED TIME= 00:00:00
DSNU010I   DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Other ways to obtain a point of consistency include using:

- The point of a SHRLEVEL REFERENCE incremental copy
- The point of an ARCHIVE LOG MODE(QUIESCE)
  
  You may use the keyword TOLOGPOINT LASTARCHQ.
- A DB2 shutdown with spaces in a consistent state
  
  You may use the keyword TOLOGPOINT LASTSHUTDOWN.
- The start of the first member of a data sharing group to be started, assuming the spaces were in a consistent state and all members were down
- RECOVERY MANAGER for DB2 to locate quiet points based on log ranges
Log Master to locate quiet points in the log

2 Recover with OUTCOPY ONLY using TORBA (or TOLOGPOINT).

The point of consistency, which in this case is the quiesce point indicated by TORBA LASTQUIESCE, is now used to create new copies from the old copies and logs without reading the space itself.

The example the following figure uses:

- The dynamic allocation of output data sets with wildcards for defining the data set name
- The automatic generation of GDGs using the AFRGDG DD statement
- The recovery of all indexes associated with each table space that is being recovered

Figure 187: Example JCL to create copies using dynamic allocation with automatic generation of GDGs

```
//RECOVER EXEC PGM=AFRMAIN,REGION=0M,
// PARM='DBAJ,AFRAQDN,NEW,MSGLEVEL(1)' 
//STEPLIB DD DISP=SHR,DSN=product.libraries
// DD DISP=SHR,DSN=DB2.DSNEXIT
// DD DISP=SHR,DSN=DB2.DSNLOAD
//AFRGDG DD DSN=AFR.PROD.GDGDEF(GDGDEF01),DISP=SHR
//SYSIN DD *
//OPTIONS OUTCOPY ASCODED INDEXLOG YES
OUTPUT COPY DSNAME AFR.IC.&DB.&SP(+1) UNIT CART CATLG YES STACK YES
RECOVER TABLESPACE AFRIN.AFRACCT
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
   INDEX ALL TABLESPACE AFRIN.AFRACCT
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
TABLESPACE AFRIN.AFRTRAN
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
   INDEX ALL TABLESPACE AFRIN.AFRTRAN
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
TABLESPACE AFRIN.AFRSUMM
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
   INDEX ALL TABLESPACE AFRIN.AFRSUMM
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
TABLESPACE AFRIN.AFRHIST
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
   INDEX ALL TABLESPACE AFRIN.AFRHIST
   OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
TORBA LASTQUIESCE
```
Example 2: Creating an image copy without accessing the space

You can create a new image copy of a table space or index at any time by running a RECOVER command with OUTCOPY ONLY.

The underlying space will not be accessed or affected in any way.

- If you do not specify a log point (with TORBA or TOLOGPOINT), the log is used up to the point where the utility begins. The log information is combined with any copies or change accumulation files to create a new full copy.

- If you do specify a log point (with TORBA or TOLOGPOINT), the log is used up to the target LRSN. The log information is combined with any copies or change accumulation files to create a new full copy.

Whether or not you select a TORBA or TOLOGPOINT, NGT Recover registers the copy as SHRLEVEL CHANGE unless the log point is a point of consistency.

Example JCL to create the new image copy and use dynamic allocation for the output image copy is shown in the following figure.

Figure 188: Example NGT Recover JCL to create new image copy using dynamic allocation

```plaintext
//RECOVER EXEC PGM=AFRMAIN,REGION=0M,  
// PARM='DBAN,AFRAQDN,NEW,MSGLEVEL(1)'  
//STEPLIB DD DISP=SHR,DSN=product.libraries  
// DD DISP=SHR,DSN=AFR.DSNEXIT  
// DD DISP=SHR,DSN=AFR.DSNLOAD  
//AFRGDG DD DSN=AFR.PROD.GDGDEF(GDGDEF01),DISP=SHR  
//SYSIN DD *  
OPTIONS OUTCOPY ASCODED INDEXLOG YES  
OUTPUT COPY DSNAME AFR.IC.&DB.&SP(+1)  
UNIT SYSDA CATLG YES MODELDCB SYS1.MODEL SPACE (10,10) CYL  
RECOVER TABLESPACE AFRFIN.AFRACCT  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY  
TABLESPACE AFRFIN.AFRTRAN  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY  
TABLESPACE AFRFIN.AFRSUMM  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY  
TABLESPACE AFRFIN.AFRHIST  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY  
INDEX AFRFIN.AFRACCTI  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY  
INDEX AFRFIN.AFRTRANI  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY  
INDEX AFRFIN.AFRSUMMI  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY  
INDEX AFRFIN.AFRHISTI  
OUTCOPY ONLY REGISTER ALL OUTCOPYDDN COPY
```
Example 2: Creating an image copy without accessing the space
NGT Recover installation options

The BMC Next Generation Technology Recover for DB2 for z/OS product is installed by using the Installation System from BMC. During the installation, the customization process generates a customized installation data set.

This data set contains customized jobs that install NGT Recover into your specific IBM DB2 environment. One of these jobs, $C30DOPT, establishes the default processing option values that NGT Recover uses.

The $C30DOPT job assembles the options macro. The macro contains the NGT Recover processing options and the values for those options that are shipped with NGT Recover. When the Installation System-generated customization job is submitted, it links the AFR$OPTS installation options module in the APF-authorized library that is designated by your site. If any values for these options are changed during customization, the new values override the values from the options macro.

You can customize the installation of NGT Recover by changing the values for the NGT Recover installation options. However, if you change any of the values in $C30DOPT after NGT Recover has been installed, you must rerun the jobs for these changes to take effect.

You can also create additional options modules that allow you to use different values of these options for different executions of NGT Recover. For example, you might use the default installation options module for most jobs but create another options module with customized values for certain options for special situations. For information about specifying an options module at runtime, see “Building and running NGT Recover jobs” on page 311. For more information about customizing your installation of NGT Recover, see the Installation System Reference Manual and the BMC Products and Solutions for DB2 Customization Guide.

Installation options macro listing

The following figure shows the installation options macro listing for NGT Recover.
Quick reference of NGT Recover installation options

This section describes the NGT Recover installation options including their default values.
For quick reference, the following table presents the options in alphabetical order and includes their default values, a brief description, and a reference to more information.

### Table 28: NGT Recover installation options

<table>
<thead>
<tr>
<th>NGT Recover installation option</th>
<th>Default value a</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“AMSCAT=NO” on page 578</td>
<td>NO</td>
<td>Defines use of catalog option in the IDCAMS define</td>
</tr>
<tr>
<td>“AUTOSIZE=YES” on page 578</td>
<td>YES</td>
<td>Specifies dynamic sizing for output image copies or change accumulation files</td>
</tr>
<tr>
<td>“AUX=NO” on page 578</td>
<td>NO</td>
<td>Specifies if the auxiliary objects related to a base table space are to be included in recovery, and for DB2 Version 10 and later, if the history table related to a specified system-maintained temporal table is included in the recovery</td>
</tr>
<tr>
<td>“BACKOUT=AUTO” on page 580</td>
<td>AUTO</td>
<td>Specifies the backout recovery option</td>
</tr>
<tr>
<td>“BINDQUALIFIER=AFRvvr” on page 580</td>
<td>AFRvvr</td>
<td>NGT Recover bind qualifier, for example AFR111</td>
</tr>
<tr>
<td>“CHECKINT=0” on page 580</td>
<td>0</td>
<td>Interval between checkpoints in minutes</td>
</tr>
<tr>
<td>“CHECKPT=PHASE” on page 581</td>
<td>PHASE</td>
<td>Checkpoint for restart</td>
</tr>
<tr>
<td>“CPUMIPS=200” on page 581</td>
<td>200</td>
<td>Rate that a CPU can execute instructions in millions of instructions per second (MIPS)</td>
</tr>
<tr>
<td>“DATACLAS” on page 581</td>
<td>None</td>
<td>SMS data class</td>
</tr>
<tr>
<td>“DATAMVR=” on page 582</td>
<td>None</td>
<td>Provides XBM with the name of the program to use to copy a data set if the data set is not on snappable DASD</td>
</tr>
<tr>
<td>“DEFINE=NO” on page 582</td>
<td>NO</td>
<td>Specifies if DEFINE NO objects will be instantiated</td>
</tr>
<tr>
<td>“DISKIORATE=100” on page 582</td>
<td>100</td>
<td>Number of megabytes per second that NGT Recover can read from disk</td>
</tr>
<tr>
<td>“EATTR=” on page 582</td>
<td>None</td>
<td>Enable extended attributes to allocate an extended format sequential data set (supported by IBM z/OS Version 1.11 and later)</td>
</tr>
<tr>
<td>“ERRCONT=10” on page 583</td>
<td>10</td>
<td>Maximum severe errors</td>
</tr>
<tr>
<td>“FCPPRC=NONE” on page 583</td>
<td>NONE</td>
<td>Controls what happens if you specify SNAP=VSAM and the data sets are on disk that is capable of IBM FlashCopy</td>
</tr>
<tr>
<td>NGT Recover installation option</td>
<td>Default value</td>
<td>Brief description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>“HISTORY=YES” on page 584</td>
<td>YES</td>
<td>Use BMC HISTORY table</td>
</tr>
<tr>
<td>“INDEXLOG=NO” on page 584</td>
<td>NO</td>
<td>Specifies RECOVER INDEX behavior</td>
</tr>
<tr>
<td>“IXRECP=NO” on page 585</td>
<td>NO</td>
<td>Puts indexes in RECP status if TS PIT recovery</td>
</tr>
<tr>
<td>“KEYDSNAM= keyDatasetName” on page 586</td>
<td>None</td>
<td>Name of the key data set for encryption</td>
</tr>
<tr>
<td>“KSORTSHARE=YES” on page 586</td>
<td>YES</td>
<td>Specifies whether key sorts are shared, which determines number of active key sorts (with MAXKSORT)</td>
</tr>
<tr>
<td>“LOCCASEL=(LP,LB)” on page 586</td>
<td>(LP,LB)</td>
<td>Resource selection for local site change accumulation files</td>
</tr>
<tr>
<td>“LOCCPSEL=(FC,LP,LB)” on page 586</td>
<td>(FC,LP,LB)</td>
<td>Resource selection for local site copies</td>
</tr>
<tr>
<td>“MAXDRIVE=0” on page 587</td>
<td>0</td>
<td>Specifies maximum tape drives to be used</td>
</tr>
<tr>
<td>“MAXLOGS=5” on page 588</td>
<td>5</td>
<td>Specifies the maximum number of logs to allocate</td>
</tr>
<tr>
<td>“MAXLSORT=0” on page 588</td>
<td>0</td>
<td>Specifies the maximum number of log sorts that can run concurrently and determines the number of MERGE/RESTORE/SNAP phases that can run in parallel whether or not log records are processed</td>
</tr>
<tr>
<td>“OPNDB2ID=YES” on page 589</td>
<td>YES</td>
<td>Specifies the use of the DB2 RACF ID</td>
</tr>
<tr>
<td>“OUTCOPY=ASCODED” on page 589</td>
<td>ASCODED</td>
<td>Specifies how to create output image copies</td>
</tr>
<tr>
<td>“PLANRECV=AFRBvvr” on page 590</td>
<td>AFRBvvr</td>
<td>NGT Recover main plan name, for example AFRBvvr</td>
</tr>
<tr>
<td>“PUBLICPLAN=YES” on page 590</td>
<td>YES</td>
<td>Grants PUBLIC privilege to run NGT Recover</td>
</tr>
<tr>
<td>“RCLTSK=10” on page 590</td>
<td>10</td>
<td>Specifies the maximum recall tasks</td>
</tr>
<tr>
<td>“RDB2STAT=YES” on page 590</td>
<td>YES</td>
<td>Restores original status of spaces</td>
</tr>
<tr>
<td>“REMCASEL=(RP,RB)” on page 590</td>
<td>(RP,RB)</td>
<td>Resource selection for remote site change accumulation files</td>
</tr>
<tr>
<td>“REMCPSEL=(RP,RB,FC)” on page 591</td>
<td>(RP,RB,FC)</td>
<td>Resource selection for remote site copies</td>
</tr>
<tr>
<td>“RESINV=0K” on page 591</td>
<td>0K</td>
<td>Specifies memory reserves for IDCAMS</td>
</tr>
<tr>
<td>“RICHK=NO” on page 591</td>
<td>NO</td>
<td>Specifies if NGT Recover will check for referential integrity constraints on the objects being recovered</td>
</tr>
</tbody>
</table>
### Descriptions of Installation Options

This topic describes the basic NGT Recover installation options and their default values.

<table>
<thead>
<tr>
<th>NGT Recover installation option</th>
<th>Default value&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“SMCORE=(0K,0K)” on page 591</td>
<td>(0K,0K)</td>
<td>Specifies memory for sort</td>
</tr>
<tr>
<td>“SNAP=HW” on page 592</td>
<td>HW</td>
<td>Indicates if you want NGT Recover to recover using VSAM copies, even if the data set is not on a snappable disk</td>
</tr>
<tr>
<td>“SORTDEVT” on page 593</td>
<td>Determined by BMCSORT installation options</td>
<td>Specifies device type for sort work space</td>
</tr>
<tr>
<td>“SORTNUM” on page 593</td>
<td>Determined by BMCSORT installation options</td>
<td>Specifies number of allocated sort work space DSNs</td>
</tr>
<tr>
<td>“STORCLAS” on page 594</td>
<td>none</td>
<td>SMS storage class</td>
</tr>
<tr>
<td>“TBUFFS=1000” on page 594</td>
<td>1000</td>
<td>Specifies buffer manager pages</td>
</tr>
<tr>
<td>“TZRULE=NONE” on page 594</td>
<td>NONE</td>
<td>Specifies the rule that determines when daylight savings time (DST) begins and ends (Use this installation option to enable DST adjustments.)</td>
</tr>
<tr>
<td>“USEHDROBIDS=YES” on page 596</td>
<td>YES</td>
<td>Indicates if the OBIDs in the header are valid or not.</td>
</tr>
<tr>
<td>“USELOGS=(ACT1,ACT2,ARC1,ARC2)” on page 597</td>
<td>(ACT1,ACT2, ARC1,ARC2)</td>
<td>Specifies the order in which to read active and archive logs</td>
</tr>
<tr>
<td>“WKUNIT=SYSALLDA” on page 597</td>
<td>SYSALLDA</td>
<td>Indicates unit to dynamically allocate work files</td>
</tr>
<tr>
<td>“WTOR=YES” on page 598</td>
<td>YES</td>
<td>Indicates to issue WTOR if NGT Recover finds spaces in STOPP status</td>
</tr>
<tr>
<td>“XBMID=ssid or xbmGroup” on page 598</td>
<td>None</td>
<td>Specifies the XBM subsystem ID or XBM group name for Instant Snapshots</td>
</tr>
<tr>
<td>“ZIIP=ENABLED” on page 599</td>
<td>ENABLED</td>
<td>Determines whether zIIP processing is enabled</td>
</tr>
</tbody>
</table>

<sup>a</sup> The default values for the installation options are those generated through the Installation System. The installation system enables you to customize the default values for your site and saves them in the installation system profile variables.
**AMSCAT=NO**

This option indicates whether to include the CATALOG option in IDCAMS DEFINE CLUSTER commands when defining data sets for STOGROUP-defined spaces. The CATALOG option specifies the VCAT name specified in the STOGROUP definition. AMSCAT=NO is the default and does not include the CATALOG option. AMSCAT=YES includes the CATALOG option.

**AUTOSIZE=YES**

This option turns dynamic sizing for output image copies or change accumulation output files on or off.

Valid values are YES, Y, NO, and N. The default value is YES. AUTOSIZE=YES means dynamic sizing of output image copies or change accumulation output files to DASD occurs. AUTOSIZE=NO means output image copies or change accumulation output files to DASD are allocated using the primary and secondary quantities that are specified in the R+/CHANGE ACCUM repository.

You can override the AUTOSIZE installation option by using the AUTOSIZE option on the OPTIONS command (see “AUTOSIZE” on page 138).

**AUX=NO**

Use the AUX installation option to specify if the auxiliary objects related to a base table space are to be included in recovery.

Valid values are NO, ALL, YES, XML, LOB, HISTORY, and ARCHIVE. The default value is NO.

These values have the following meanings:

- **AUX=NO** does not include any auxiliary objects in the recovery.
- **AUX=ALL** includes XML and LOB objects in the recovery. It does not include HISTORY spaces with their related auxiliary objects when a table space with system-maintained temporal tables is specified in SYSIN.
- **AUX=YES** includes both LOB and XML objects in the recovery, as well as history spaces when a table space with system-maintained temporal tables is specified in SYSIN.
- **AUX=XML** includes all XML data objects along with the XML base table space. If the table space specification is found on a RECOVER TABLESPACE or SIMULATE TABLESPACE statement, all implicitly created XML table spaces will be processed along with the base XML table space. If the table space specification is found on a RECOVER INDEX(ALL), REBUILD INDEX(ALL), or SIMULATE
INDEX(ALL) statement, all implicitly created XML and node ID indexes on the XML base table space as well as all explicitly created non-LOB indexes will be processed.

- **AUX=LOB** includes all LOB data objects along with the LOB base table space. If the table space specification is found on a RECOVER TABLESPACE or SIMULATE TABLESPACE statement, all table spaces created with the LOB attribute will be processed along with the base LOB table space. If the table space specification is found on a RECOVER INDEX(ALL), REBUILD INDEX(ALL) or SIMULATE INDEX(ALL) statement, all indexes on the LOB base table space and all non-LOB explicitly created indexes will be processed.

- **AUX=HISTORY** includes history spaces when a table space with system-maintained temporal tables is specified in SYSIN. Similar to DB2 RECOVER Version 10 VERIFYSET NO behavior, NGT Recover does *not* check for the following:

  - That the system-maintained temporal space and the history space are recovered as a set
  - That the recovery of the system-maintained temporal space and the history space is to the same log point
  - That all DSNUMs are recovered in the same SYSIN

- **AUX=ARCHIVE** tells NGT Recover whether to recover archive tables if you have specified table spaces with archive-enabled base tables in SYSIN. This option does not include related XML or LOB objects.

You can override the value of the AUX installation option at runtime by using the AUX option on the OPTIONS command (“AUX” on page 117) or in the table space specification for any of the following NGT Recover statements:

- **RECOVER TABLESPACE**
- **RECOVER INDEX(ALL) TABLESPACE**
- **RECOVER INDEXSPACE(ALL) TABLESPACE**
- **REBUILD INDEX(ALL) TABLESPACE**
- **LOGSCAN TABLESPACE**
- **LOGSCAN INDEX(ALL) TABLESPACE**
- **LOGSCAN INDEXSPACE(ALL) TABLESPACE**

**Note**

NGT Recover ignores the AUX option when you specify OBJECTSET. Also, the AUX option has no effect for the ACCUM TABLESPACE, RECOVER UNLOADKEYS, and RECOVER BUILINDEX commands.
When you specify AUX=YES, AUX=ALL, AUX=XML, AUX=LOB, AUX=HISTORY, or AUX=ARCHIVE, you cannot specify the following options:

You can specify the following options with AUX *only* if using AUX=NO:

- **TOCOPY dataSetName**
- **OBIDXLAT** without defaults
- **OUTCOPYDSN**
- **RECOVERYDSN**

## BACKOUT=AUTO

This option specifies the default for BACKOUT recovery. Valid values are AUTO and NO. AUTO is the default value. With this default value, if you are running with a Recovery Management password, any point-in-time recovery uses BACKOUT AUTO. If you are not using a Recovery Management password, point-in-time recoveries use a standard forward recovery.

You can override the value of the BACKOUT installation option at runtime by using the BACKOUT option on the OPTIONS command ("BACKOUT" on page 122).

For more information about BACKOUT AUTO recovery, see “Using BACKOUT AUTO recovery” on page 47.

If you do not want BACKOUT AUTO to be the default value, change the BACKOUT installation option to BACKOUT NO.

If the site type in DSNZPARM is a recovery site, NGT Recover uses BACKOUT NO as the default value. If you are doing disaster recovery at local site (site type is local site in DSNZPARM), you should set the BACKOUT installation option to NO for disaster recovery.

## BINDQUALIFIER=AFRvvr

Use the BINDQUALIFIER installation option to specify the NGT Recover bind qualifier for the dynamic bind process.

The bind qualifier determines which set of synonyms NGT Recover is to use. The BINDQUALIFIER value can have a maximum length 16 bytes. The default value is AFRvvr, where v represents a two-character version number and r represents a one-character release number, such as AFR111.

## CHECKINT=0

This option specifies the time in minutes between checkpoints. This option allows you to balance the frequency of restart points with the overhead incurred when
taking restart points. If CHECKPT=PHASE, a restart point is taken only if more minutes than those specified by CHECKINT have elapsed since the last restart point was taken. The default interval is 0 minutes. Use CHECKINT=0 and CHECKPT=PHASE to force a restart checkpoint after every phase.

You can override the CHECKINT installation option by using the CHECKINT keyword with the OPTIONS command (see “CHECKINT (integer)” on page 112).

**CHECKPT=PHASE**

This option specifies whether restart checkpoints are taken.

Valid values are NO and PHASE. The default value is PHASE.

CHECKPT=NO indicates that no checkpoints are taken. CHECKPT=PHASE specifies that checkpoints are taken at the end of each phase. You can override the CHECKPT installation option by using the checkpoint override parameter (see “Checkpoint override parameter (checkPt)” on page 316). Also, see the information about the CHECKINT option (which further limits the frequency of taking checkpoints) that follows.

**CPUMIPS=200**

Use CPUMIPS to indicate the rate that a CPU can execute instructions in millions of instructions per second (MIPS). The default value is 200.

*Note*

CPUMIPS is used to derive many of the performance factors for estimation. Estimation requires a BMC Recovery Management for DB2 solution password. For more information about estimation, see the Recovery Management for DB2 User Guide.

**DATACLAS**

Specify DATACLAS= when you want to provide an SMS data class name.

The value of name must be a valid SMS data class name, not exceeding eight characters.

NGT Recover forces CATLG=YES when you specify DATACLAS.

By using the DATACLAS syntax option in an OUTPUT statement at runtime, you can override the value set during installation.
**DATAMVR=**

The DATAMVR installation option provides XBM with the name of the program to use to copy a data set if a data set snap fails. To use DFDSS as the data mover, specify DATAMVR=ADRDSSU.

You can override the DATAMVR installation option by using the DATAMVR parameter with the OPTIONS command (see “DATAMVR programName” on page 129).

**DEFINE=NO**

The DEFINE option specifies if DEFINE NO objects will be instantiated. Valid values are YES and NO. The default value is NO. If you specify YES, NGT Recover instantiates DEFINE NO objects if necessary, and updates the catalog and DBD to reflect that the object has been defined. If you specify NO, NGT Recover does not process DEFINE NO objects.

You can override the DEFINE installation option by using the DEFINE keyword with the OPTIONS command (see “DEFINE” on page 121) or in the options for the RECOVER or IMPORT commands.

**DISKIORATE=100**

Use DISKIORATE to specify the number of megabytes per second that NGT Recover can read from disk. The default value is 100.

*Note*

DISKIORATE is used to derive many of the performance factors for estimation. Estimation requires a BMC Recovery Management for DB2 solution password. For more information about estimation, see the *Recovery Management for DB2 User Guide*.

**EATTR=**

Use EATTR to specify whether a data set supports extended attributes or not. If EATTR is not specified, which is the default, an SMS DATAACLAS can provide the value.

You cannot use an image copy made to the cylinder-managed portion of an extended address volume (EAV) under z/OS Version 1.11 on z/OS Version 1.10 because z/OS Version 1.10 does not support sequential data sets in the cylinder-managed portion of an EAV.

z/OS Version 1.11 or later supports the EATTR option. For earlier versions of z/OS, you must set EATTR= (or EATTR=None).
You can set EATTR to OPT or NO in the JCL.

When you create an OUTCOPY (OUTCOPY specification on page 193) and want to force the copy to the cylinder-managed space of an EAV, specify the SPACE keyword.

Valid values for EATTR are:

- Specifying no value for EATTR (EATTR=), the default, allows the value for EATTR to be set by an SMS DATACLAS. (EATTR= is the same as specifying EATTR=NONE.)
  Using the default value allows you to have your environment set up to use extended attributes.

- OPT specifies that extended attributes are optional for the data set.
  You must set EATTR=OPT to allocate an extended format sequential data set. By using EATTR=OPT, NGT Recover supports sequential data sets in the cylinder-managed portion of EAVs.
  If you specify EATTR=OPT, NGT Recover specifies the EATTR attribute when it dynamically allocates the output data set and overrides the EATTR option in the SMS DATACLAS, if one exists.
  Extended format sequential data sets must be allocated on SMS-managed volumes and the size of the data set must be greater than the EAV break point, which is typically 10 cylinders.

- NO specifies that the data set cannot have extended attributes.
  If you specify EATTR=NO, NGT Recover specifies the EATTR attribute when it dynamically allocates the output data set and overrides the EATTR option in the SMS DATACLAS, if one exists.

By using the EATTR syntax option in an OUTPUT statement at runtime, you can override the value set during installation. (See “EATTR” on page 148.)

**ERRCONT=10**

This option specifies the maximum number of times to continue processing after encountering a severe error. ERRCONT + 1 errors could occur before NGT Recover terminates.

**FCPPRC=NONE**

Use FCPPRC to control what happens if you specify SNAP=VSAM, and the data sets are on disk that is capable of FlashCopy.
The following values are valid for the FCPPRC installation option when the target volume is a primary in a PPRC mirror pair:

- NONE (the default) specifies that no FCTOPPRCPRIMARY option is passed to DFDSS.
- PREP specifies that preserving the mirror during the FlashCopy is preferred.
- REQ specifies that preserving the mirror is required.
- NO specifies that the mirror is not preserved during a FlashCopy.

**HISTORY=YES**

The HISTORY option specifies whether NGT Recover is to update the BMCHIST table. Valid values are YES and NO. The default value is YES. If you specify YES, NGT Recover updates the BMCHIST table by writing one row per NGT Recover job step. If you specify NO, NGT Recover bypasses any updates to the BMCHIST table.

See “BMCHIST table” on page 611 for more information about the BMCHIST table.

**INDEXLOG=NO**

This option specifies what behavior RECOVER INDEX has. Valid values are NO, YES, and AUTO.

- INDEXLOG=NO, the default, indicates that RECOVER INDEX and REBUILD INDEX are synonyms and indexes are rebuilt from the keys extracted from the table space.
- INDEXLOG=YES indicates that RECOVER INDEX recovers indexes from copies, the log, or both, while REBUILD INDEX rebuilds indexes from the keys extracted from the table space.
- INDEXLOG=AUTO indicates that RECOVER INDEX first attempts an index recovery from image copies, the log, or both. However, if an image copy for the index does not exist, NGT Recover converts the request to a REBUILD INDEX request.

INDEXLOG=AUTO supports BACKOUT recoveries. However, conversion from RECOVER INDEX to REBUILD INDEX is not supported for following options:

- OUTCOPY: REBUILD INDEX does not include the OUTCOPY option so this request is not eligible for conversion.
- OBIDXLAT: REBUILD INDEX does not include the OBIDXLAT option so this request is not eligible for conversion.
— TOLOGPOINT: If the RECOVER INDEX request includes the TOLOGPOINT option, the request is eligible for conversion to REBUILD INDEX only if the associated table space is also recovered in the same job step.

— TOCOPY: If the RECOVER INDEX request includes the TOCOPY option, the request is eligible for conversion to REBUILD INDEX only if the associated table space is also recovered in the same job step.

— DSNUM: If the RECOVER INDEX command statement was for a specific data set of a nonpartitioned index, the request is not eligible for conversion to REBUILD INDEX.

**Note**
If you plan to make index image copies, you might consider using INDEXLOG=YES or INDEXLOG=AUTO, but realize that you may have to change some RECOVER INDEX command statements to REBUILD INDEX. However, if you have no immediate plans to start making index image copies, you may prefer to leave the default, INDEXLOG=NO, so that you will not have to change syntax.

You can override the INDEXLOG installation option by using the INDEXLOG keyword with the OPTIONS command (see “INDEXLOG” on page 124).

**Note**
The RECOVER INDEXSPACE command statement always invokes a recovery based on copies and logs, regardless of the INDEXLOG setting. Also, REBUILD INDEX always rebuilds indexes, regardless of the setting for INDEXLOG.

**IXRECP=NO**

IXRECP specifies whether NGT Recover alerts you that indexes have not been recovered. Valid values are NO and YES. IXRECP=YES tells NGT Recover to issue a warning message for each index that has not been recovered with the table space, and to put each such index in RECP status. IXRECP=NO, the default, disables this feature.

You can override the IXRECP installation option by using the IXRECP keyword with the OPTIONS command (see “IXRECP” on page 126).

**Note**
Use this option when a point-in-time recovery of a table space is performed. It is important that any indexes on the space be recovered so that index data is synchronized with the data to which it refers.
KEYDSNAM= keyDataSetName

The KEYDSNAM option specifies the name of the key data set that is used for encrypted copies.

For more information about encrypted copies and the key data set, see “Recovering encrypted copies” on page 506.

Note

Encryption is a feature of the BMC Recovery Management for DB2 solution and requires a valid Recovery Management solution password.

KSORTSHARE=YES

This option specifies if key sorts are shared among NGT Recover table space recoveries (MERGE phases) running in parallel.

When you specify YES, which is the default value, NGT Recover uses up to the value specified for MAXKSORT active key sorts at any given time. If sufficient key sorts are not available when a table space recovery begins execution, keys will be obtained later by an UNLOAD phase.

When you specify NO, each MERGE phase has its own set of key sorts and up to MAXKSORT * MAXLSORT key sorts can be active at any given time. Since the number of sorts that can be active in a system is fairly small—usually no more than 30—a value of NO for this option may severely limit the number of recovery operations that NGT Recover can perform in parallel when index rebuilds are also requested.

You can override the KSORTSHARE installation option by using the KSORTSHARE parameter on the OPTIONS command (see “KSORTSHARE” on page 137).

LOCCASEL=(LP,LB)

This option specifies the sequence of change accumulation files for selection for local site recovery. The default sequence is local primary (LP), and then local secondary (LB). You can specify a pair of values from LP, LB, remote primary (RP), and remote secondary (RB) in any order.

LOCCPSEL=(FC,LP,LB)

This option specifies the sequence of image copy files for selection for local site recovery. The default sequence is IBM FlashCopy (FC), local primary (LP), then local secondary (LB). You can specify two or three values from FC, LP, LB, system backup (SB), remote primary (RP), and remote secondary (RB) in any order.
**MAXDRIVE=0**

This option specifies the default value for the maximum number of tape drives to use during a utility run. The default value is 0, which means tape drive usage is unlimited.

You can override the MAXDRIVE installation option by using the MAXDRIVES keyword with the OPTIONS command (see “MAXDRIVES integer” on page 115).

**MAXKSORT**

This option specifies the maximum number of index key sorts that can run in parallel.

The value of MAXKSORT also determines the concurrency of index rebuilds. Thus, MAXKSORT can be used to improve recovery performance. (The KSORTSHARE option also impacts index recovery. For more information, see “KSORTSHARE=YES” on page 586.)

---

**Note**

When you rebuild the indexes of a multi-data-set, nonpartitioned table space, the UNLOADs run serially in the main task, but the REBUILDs are multitasked.

Valid values are 1 to 999. The default value is given by the following formula:

\[ \text{minimum}(2 \times \text{the number of CPUs}, 12) \]

---

**Note**

Each sort requires about 256 KB of memory below the line. Values for MAXKSORT greater than 12 are not recommended.

You can override the MAXKSORT installation option by using the MAXKSORT parameter with the OPTIONS command. For more information, see “Overview of the UNLOADKEYS and BUILDINDEX strategy” on page 463.

When you specify MAXKSORT, the following files are dynamically allocated if you do not code them in JCL:

- SYSOU\[nnn\]: sort message files
  - \[nnn\] is a number between 1 and the value that is specified for MAXKSORT.
SxxxWKnn: key sort work files

xxx is a number between 1 and the value that is specified for MAXKSORT. nn is a number between 1 and the value that is specified for the SORTNUM installation option or the OPTIONS SORTNUM parameter.

When you use dynamic allocation for these files, NGT Recover determines the optimal number of files to use.

Note

When the value of MAXKSORT is greater than 1, NGT Recover ignores the WORKDDN option on the REBUILD INDEX command and issues a warning message. To use the WORKDDN option, specify a value of 1 for MAXKSORT. For a description of WORKDDN, see “WORKDDN DDName” on page 227.

MAXLOGS=5

This option specifies the default value for the maximum number of log files that NGT Recover reads at one time. MAXLOGS must be greater than 0. The default value is 5.

You can override the MAXLOGS installation option by using the MAXLOGS keyword with the OPTIONS command (see “MAXLOGS integer” on page 115).

MAXLSORT=0

This option specifies the maximum number of log sorts that can run concurrently and also determines the number of MERGE/RESTORE/SNAP phases that can run in parallel, whether or not log records are processed. You use MAXLSORT to improve recovery performance.

Valid values are 0 to 999. When MAXLSORT=0, the default value is determined by NGT Recover.

Note

BMC suggests values from 10 to 12 for MAXLSORT.

You can override the MAXLSORT installation option by using the MAXLSORT parameter on the OPTIONS command.

When you specify MAXLSORT, the following files are dynamically allocated if you do not code them in JCL:
LOGOU\[n\[n\]n: sort message files

\[n\]n is the number of the log sort and is a number between 1 and the value that is specified for MAXLSORT.

Lxx\[W\[n\[n\]n: sort work files

\[x\]x is the number of the log sort and is a number between 1 and the value that is specified for MAXLSORT. \[n\]n is the number of the work data set. For example, if MAXLSORT=3 and two sort work files are required for each sort, the DDs would be specified as follows:

- \L001WK01 DD...
- \L001WK02 DD...
- \L002WK01 DD...
- \L002WK02 DD...
- \L003WK01 DD...
- \L003WK02 DD...

When you use dynamic allocation for these files, NGT Recover determines the optimal number of files to use.

**OPNDB2ID=YES**

This option indicates whether to use the DB2 RACF ID instead of the RACF ID of the user running NGT Recover when opening the DB2 data sets, such as table spaces and logs. Valid values are YES and NO. The default value is YES.

OPNDB2ID=YES uses the RACFID of DB2. OPNDB2ID=NO uses the RACFID of the user running NGT Recover. If you specify OPNDB2ID=NO, the user must have the appropriate RACF authority.

If the RACF started procedures table (ICHRIN03) specifies DB2 as a privileged or trusted task and has no user ID associated with the DB2 address space, you cannot use OPNDB2ID to allow NGT Recover to access the DB2 data sets. In this case, the user running NGT Recover must have RACF authority to access the data sets needed for recovery.

*Note*

OPNDB2ID works under data sharing only if all of the RACF IDs for the members of a group are the same. The authorizations for the BSDS and log data sets must also be the same.

For any security system other than RACF, NGT Recover ignores this option and uses the ID of the user running NGT Recover.

**OUTCOPY=ASCODED**

This option specifies how to create output image copies on partitioned table spaces and indexes. Valid values are ASCODED and BYPART. If you specify
OUTCOPY=ASCODED, the default, and the recovery specifies DSNUM ALL, NGT Recover creates a DSNUM 0 image copy. If a recovery is by DSNUM and you specify BYPART or ASCODED for this option, NGT Recover creates the image copy by DSNUM.

You can override the OUTCOPY installation option by using the OUTCOPY keyword with the OPTIONS command (see “OUTCOPY” on page 116).

**PLANRECV=AFRB ννr**

This option is the NGT Recover execution plan. The default value is AFRB ννr, where νν represents a two-character version number and r represents a one-character release number, such as AFRB111.

**PUBLICPLAN=YES**

Use the PUBLICPLAN installation option to grant the PUBLIC privilege to run NGT Recover. Valid values for PUBLICPLAN are YES and NO. The default value is YES. If you set the value of this option to NO, the installation process will not do any grants, which means that you must grant execute authority to users as needed.

**RCLTSK=10**

This option specifies the maximum number of concurrent tasks that are used to recall archived image copies and log data sets when you use the EARLYRECALL option. RCLTSK=0 disables EARLYRECALL. The default value is 10.

For more information about EARLYRECALL, see “EARLYRECALL / NOEARLYRECALL” on page 107.

**RDB2STAT=YES**

This option specifies whether the original status of the recovered objects is to be restored after a successful recovery. Valid values are YES and NO. The default value is YES. RDB2STAT=NO leaves the spaces stopped. RDB2STAT=YES restores the original status.

You can override the RDB2STAT installation option by using the RDB2STAT override parameter (see “RDB2STAT override parameter (rdb2Stat)” on page 317).

**REMCASEL=(RP,RB)**

This option specifies the sequence of change accumulation files for selection for remote site recovery. The default sequence is remote primary (RP), then remote
secondary (RB). You can specify a pair of values from RP, RB, local primary (LP), and local secondary (LB) in any order.

**REMCPSEL=(RP,RB,FC)**

This option specifies the sequence of image copy files for selection for remote site recovery. The default sequence is remote primary (RP), remote secondary (RB), and then IBM FlashCopy (FC). You can specify two or three values from RP, RB, FC, local primary (LP), and local secondary (LB) in any order.

**RESINV=0K**

RESINV specifies the amount of memory reserved below the 16-MB line by the sort routine to allow for IDCAMS processing. The default value of RESINV=0K allows the sort routine defaults to be used. *BMC recommends that you use the default value so that the sort routine uses its algorithms effectively.*

**RICHK=NO**

The RICHK option specifies if NGT Recover will check for referential integrity constraints on the objects being recovered. Valid values are YES and NO. The default value is NO. If you specify YES, NGT Recover checks for referential integrity constraints and sets check pending (CHKP) status for dependent objects not recovered in the same step to the same PIT. If you specify NO, NGT Recover does not check for referential integrity.

You can override the RICHK installation option by specifying the RICHK option in the OPTIONS command (see “RICHK” on page 116) or in the options for the RECOVER or IMPORT commands.

**SMCORE=(0K,0K)**

This option specifies the amount of memory that you want each invocation of BMCSORT to use.

For more information, see “Managing sort performance” on page 474.

- The first value specifies the total amount of memory used both below and above the 16-MB line.
- The second value specifies only the amount of memory used below the 16-MB line.

You can enter either value as a number of kilobytes (using the suffix K) or megabytes (using the suffix M).
BMC recommends that you specify zero for both SMCORE values (as 0K or 0M) to allow NGT Recover to determine the optimal amount of storage to allocate above and below the 16-MB line based on the following criteria:

- Number of sorts to process
- Amount of memory that is available during optimization
- Value that you specified for REGION in either your JCL or system exits

For the first value, the following values are valid:

- Zero, specified as 0K or 0M, tells NGT Recover to determine the appropriate amount
- 4096 KB through 65536 KB (specified by using the K suffix) or 4 MB through 64 MB (specified by using the M suffix) tells NGT Recover to use the specified amount

For the second value, the following values are valid:

- 0K (or 0M) tells NGT Recover to determine the appropriate amount.
- 256 KB through 4096 KB (specified by using the K suffix) or 1 MB through 4 MB (specified by using the M suffix) tells NGT Recover to use the specified amount

**SNAP=HW**

Use SNAP to Indicate if you want NGT Recover to read VSAM copies, even if the data set is not on a snappable disk:

- SNAP=HW (the default) tells NGT Recover to use a hardware data set snapshot to restore an Instant Snapshot or VSAM data set.
- SNAP=VSAM tells NGT Recover to use conventional VSAM I/O to restore a VSAM data set if it is not on a snappable disk.

To read a VSAM copy with SNAP VSAM, you specify the name of the VSAM data set in an INCOPY statement, just as you would if it was an Instant Snapshot copy, using the INCOPY FULL SNAPSHOT DSNAMe *dataSetName* syntax.

SNAP VSAM also allows you to recover using a VSAM copy registered in BMCXCOPY or SYSCOPY if that copy is not on snappable disk.

You can override the value of the SNAP installation option at runtime by specifying the SNAP option on the OPTION command ("SNAP" on page 141).
SORTDEVT

This option specifies the default device type for the temporary sort work data sets that BMCSORT uses for sorting log records and index keys. If you do not specify a value, the value is obtained from the BMCSORT installation options.

For more information, see the *BMCSORT Reference Manual*.

You can override the SORTDEVT installation option by setting the SORTDEVT keyword in one of the following commands:

- OPTIONS
- RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX
- RECOVER UNLOADKEYS
- REBUILD INDEX

For more information, see *General information about SORTDEVT and SORTNUM on page 90*

SORTNUM

This option specifies the default number of temporary sort work data sets that are allocated dynamically by BMCSORT for sorting log records and index keys.

You can specify an integer value of 1 through 255. If you do not specify a value, the value is obtained from the BMCSORT installation options.

You can override the SORTNUM installation option by setting the SORTNUM keyword on one of the following commands:

- OPTIONS
- RECOVER TABLESPACE, RECOVER INDEXSPACE, or RECOVER INDEX
- RECOVER UNLOADKEYS
- REBUILD INDEX

When you specify this option, BMCSORT dynamically allocates the number of sort work files that it needs for each sort task up to the maximum that is illustrated in the following formula:

\[
\text{maximum dynamically allocated sort work files} = n - \text{preallocated sort work files}
\]

If you specify *integer* from 1 through 32, \(n\) equals 32.

If you specify *integer* greater than 32, \(n\) equals integer.
**Note**
Preallocated sort work files include sort work files that are allocated in your JCL.

For more information, see General information about SORTDEVT and SORTNUM on page 90.

**STORCLAS**

Specify STORCLAS= when you want to provide an SMS storage class name.

The value of name must be a valid SMS storage class name, not exceeding eight characters.

NGT Recover forces CATLG=YES when you specify STORCLAS.

By using the STORCLAS syntax option in an OUTPUT statement at runtime, you can override the value set during installation.

**TBUFFS=1000**

This option specifies the number of pages to allocate to the NGT Recover buffer manager. This pool of pages is allocated when the recovery uses random processes. For example, index builds use the buffer manager. BMC recommends that you modify this parameter only after consultation with your NGT Recover technical support analyst.

**TZRULE=None**

The TZRULE installation option defines the rule to determine when DST begins and ends, enabling NGT Recover to calculate the dates and times for the changes each year. Using TZRULE, you do not need to update the value.

TZRULE enables NGT Recover to adjust day and time values when you run the product after a DST change, but scan log records that are from a period before the time change, which is called scanning across a DST boundary.

The default value, NONE, indicates that your locale does not observe DST or that you do not want the product to make adjustments for DST. To enable DST adjustment, you must define day and time values as a character string in the following format:

```
TZRULE=startDayRule[/time],endDayRule[/time]
```
The first date (startDayRule) describes when the change to DST occurs, and the second date (endDayRule) describes when the change back happens. The time is optional and defaults to 02:00:00 (the default conversion time).

Use one of the following formats for startDayRule and endDayRule.

- **Mm.n.d**
  This is the standard format for most countries. Using this format, specify the month, week, and day of the week on which DST begins and ends, as follows:
  
  — The variable *m* represents the month; valid values are 1 through 12 (January through December).
  
  — The variable *n* represents the week of the month in which the day *d* occurs; valid values are 1 through 5. Week 1 is the first week in which day *d* occurs, and week 5 specifies the last *d* day in the month.
  
  — The variable *d* represents the day; valid values are 0 through 6 (Sunday through Saturday).

  In the following example, DST begins on the second Sunday of March at 2:00 a.m., and ends on the first Sunday of November at 2:00 a.m.: 

  $$\text{TZRULE=M3.2.0/2:00:00,M11.1.0/2:00:00}$$

  The following example, which uses the default conversion time of 2:00 a.m and omits the time specification, is also valid:

  $$\text{TZRULE=M3.2.0,M11.1.0}$$

- **Jn**
  Specify the Julian day *n*, where *n* is a value from 1 through 365. This format does not account for leap days; that is, in all years, including leap years, February 28 is day 59 and March 1 is day 60. You cannot explicitly refer to February 29.

  In the following example, DST begins on April 1 at 12:34:56 p.m., and ends on November 1 at 2:34:56 a.m.: 

  $$\text{TZRULE=J91/12:34:56,J305/2:34:56}$$

- **n**
  Specify the zero-based Julian day *n*, where *n* is a value from 0 through 365. In this format, leap days are counted, and you can refer to February 29.

  In the following example, DST begins on April 1 (leap years), or April 2 (nonleap years) at 12:34:56 p.m., and ends on November 1 at 2:34:56 a.m.: 

  $$\text{TZRULE=91/12:34:56,305/2:34:56}$$

### Considerations for DST adjustment

To understand DST adjustment, remember that NGT Recover uses the operating system to translate date and time values into the operating system’s store clock.
format. The operating system uses an offset based on your time zone to translate the values. For example: assume that you are in the northern hemisphere, you run NGT Recover in May (after the change to DST), and you define a scan range from 15:00 to 16:00 on February 14 (before the time change).

- If you set TZRULE to enable adjustment, the product knows that the time zone offset in May is different than the time zone offset that was used when DB2 wrote the log records in February (the difference is one hour). The product adjusts for this difference, selects log records from 15:00 to 16:00 on February 14, and prints the time values of those records in reports starting with 15:nn.

- If you do not set TZRULE to enable adjustment, the product uses the time zone offset for May. Because the offset was different in February, the product selects log records from 14:00 to 15:00 on February 14, and prints the time value of those log records in reports starting with 15:nn. The discrepancy in the time zone offset creates a discrepancy between the log records that you request and the log records that the product provides in output.

Be aware of the following additional points regarding DST adjustment and the TZRULE installation option:

- NGT Recover provides adjustment because the operating system does not account for changes in the time zone offset that have occurred in the past. The operating system always uses the current time zone offset when it converts a date and time value into store clock format.

- By default, adjustment is not enabled. To enable it, set the TZRULE installation option value.

- The options migration feature of the Installation System cannot carry over DST installation option values from a previous release of the product. You must provide a value for TZRULE if you want to enable DST adjustment.

- Adjustment affects only how the product handles the standard timestamp values that DB2 stores in all log records (using the operating system’s store clock format).

**USEHDROBIDS=YES**

Use USEHDROBIDS to indicate if the OBIDs in the header are valid or not. Valid values are YES, the default, and NO.

When you specify OBIDXLAT (OBIDXLAT specification on page 191) with INCOPY FULL SNAPSHOT TOCOPY syntax and do not specify OBIDs, NGT Recover looks at the OBIDs in the header page to determine if the OBIDs for the source and target are the same. If the OBIDs are the same, NGT Recover does not use a MERGE phase to update the OBIDs. Skipping the MERGE phase makes migration faster because it eliminates a read and write of every page of the space. (NGT Recover may still do an "optimized merge" phase to just update the header page.)
Sometimes the OBIDs in the header page are not correct. An example of when this situation might occur is when the space is copied from another object and the OBIDs in the header page are not translated correctly. If the OBIDs in the header page are not correct, NGT Recover might skip the MERGE phase when it should not, or the MERGE phase might fail because NGT Recover detects that the OBIDs in the header page do not match the OBIDs in the rest of the data. Under circumstances like these, you can use USEHDROBIDS to tell NGT Recover if the OBIDs in the header are valid or not.

You can override the USEHDROBIDS installation option by using the USEHDROBIDS parameter on the OPTIONS command (see “USEHDROBIDS” on page 119).

**USEHDROBIDS YES**

When you specify USEHDROBIDS YES, NGT Recover uses the OBIDs in the header page and skips the MERGE phase if possible. If NGT Recover does perform the MERGE phase and detects that the OBIDs are not correct, NGT Recover ends with an error message.

**USEHDROBIDS NO**

When you specify USEHDROBIDS NO, NGT Recover always does a MERGE phase and ignores the OBIDs in the header page.

**USELOGS=(ACT1,ACT2,ARC1,ARC2)**

Use USELOGS to specify the order in which to read active and archive logs. The default is (ACT1,ACT2,ARC1,ARC2).

You can override the USELOGS value at runtime by specifying the RESOURCE SELECTION LOGS syntax on the OPTION command (“RESOURCE SELECTION” on page 121).

Values for USELOGS are as follows:

- **ACT1** indicates the primary active log file.
- **ACT2** indicates the dual (secondary) active log file.
- **ARC1** indicates the primary archive log file.
- **ARC2** indicates the dual (secondary) archive log file.

**WKUNIT=SYSALLDA**

The unit for work data set dynamically allocated by NGT Recover (for example, for DSNUTILB or BMCSORT). If you do not want to use the default value, SYSALLDA,
you can change to some other appropriate generic name. Ensure that the unit specified is a valid one in your installation. Do not specify VIO for this option.

**WTOR=**\textbf{YES}

If one or more spaces in the RECOVER command statement remain in STOPP status, WTOR=YES provides the flexibility to issue a write to the operator with reply (WTOR), while WTOR=NO assumes an operator reply of CANCEL and terminates the job.

**XBMID=ssid or xbmGroup**

The XBMID option specifies the EXTENDED BUFFER MANAGER (XBM) subsystem ID (ssid) or XBM group name (xbmGroup) to use when you restore Instant Snapshot copies or when you use the zIIP redirection capability.

ssid is the unique identifier that you specified when you installed XBM. If you are using XBM in a DB2 data sharing environment, you can use the xbmGroup name in place of ssid. The xbmGroup value is the name of the XBM coupling facility group defined to the XBM subsystem.

For the zIIP redirection capability, if you specify an XBM subsystem and ZIIP ENABLED ("ZIIP" on page 128 and "ZIIP=ENABLED" on page 599) is in effect, NGT Recover attempts to use that subsystem to enable zIIP processing. If that subsystem is not available or if it is not at the correct maintenance level, zIIP processing is not enabled.

If you do not specify an XBM subsystem either with XBMID on the OPTIONS command or with the XBMID installation option, NGT Recover searches for an XBM subsystem at the appropriate maintenance level to enable zIIP processing.

For recovery using Instant Snapshots, NGT Recover does not discover the XBM subsystem. Use the XBMID option in either the installation options module or on the OPTIONS statement to specify it. Instant Snapshots are made by NGT Copy with XBM and are registered in BMCXCOPY. For more information, see the *BMC Next Generation Technology Copy for DB2 for z/OS Reference Manual*.

You can override the XBMID installation option by using the XBMID option with the OPTIONS command (see “XBMID ssid or XBMID xbmGroup” on page 127).

For specific information about valid characters for XBMID and the pattern matching capabilities of XBM, Instant Snapshots, and zIIP redirection, see the *EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide*. 
ZIIP=ENABLED

The ZIIP option tells NGT Recover whether to attempt to use IBM System z Integrated Information Processors (zIIPs). NGT Recover can use enclave service request blocks (SRBs) to enable zIIP processing automatically while running jobs. Using zIIP processing can reduce the overall CPU time for NGT Recover jobs.

Tip
When you use zIIPs, BMC recommends that you specify IIPHONORPRIORITY=YES in the IEAOPT member of SYS1.PARMLIB.

You can specify one of the following values:

- ENABLED tells NGT Recover to attempt to offload eligible processing to an available zIIP. If the zIIP is busy or not available, normal processing continues on a general-purpose processor.

- DISABLED tells NGT Recover to not attempt to use zIIP processing.

To enable and use zIIP processing with NGT Recover, you must:

- Have an installed authorized version of XBM or SUF
- Start and maintain an XBM subsystem in your environment
- Have a zIIP available in your environment

You can specify a particular XBM subsystem to use by specifying a value for the XBMID installation option or OPTIONS command XBMID option. For more information, see “XBMID=ssid or xbmGroup” on page 598 or “XBMID ssid or XBMID xbmGroup” on page 127.

XBM and SUF are licensed, installed, and maintained separately from NGT Recover. You can use either XBM or SUF, depending on the license that you have obtained:

- A license for the full version of the XBM product authorizes you to use all features of XBM.

- A license for SUF authorizes you to use only the snapshot and zIIP-processing features of XBM.

For more information about XBM and SUF, see the EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide.

You can override the value for this option by using the ZIIP command option (“ZIIP” on page 128).
BMCSORT installation and options

This appendix describes BMCSORT installation and options.

Installation overview

BMCSORT is installed automatically when you install a product or solution that invokes BMCSORT. You do not need a password to run BMCSORT.

BMCSORT requires that you specify a value for the installation option DYNALOC. For more information about the DYNALOC option, see “DYNALOC installation option” on page 601.

DYNALOC installation option

The DYNALOC installation option of BMCSORT provides information for dynamically allocating SORTWK data sets.

BMCSORT deallocates these data sets at the end of each sort. The content of the $AUPSMAC macro in $532SOFT follows, showing DYNALOC and the values that are shipped with BMCSORT.

$AUPSMAC DYNALOC=(SYSDA,3,ON,ON,6000000,3000000,3390,SC=,RETRY=(0,0)) X DYNAMIC ALLOC OPTIONS FOR SORT

The values that you specify in this macro apply to all invocations of BMCSORT. BMCSORT uses the same options module for all BMC products that invoke BMCSORT.

Table 29 on page 602 describes each parameter of the DYNALOC option. These parameters are positional. The values that you specify for these parameters should correspond to your site’s standards for any system sort routine.

BMCSORT overrides the values that you supplied if BMCSORT determines that it can complete sorting more efficiently than the specified values allow. An invoking
product’s options might also override the BMCSORT options values that you specify when one of the following conditions exists:

- The values in the invoking product’s dynamic allocation installation options or corresponding command options conflict with the values that you specify.
- You turn on BMCSORT SORTWK dynamic allocation from the product that invokes BMCSORT, and you specify OFF for the position 3 parameter.

BMCSORT dynamically allocates SORTWK files as necessary.

### Table 29: DYNALOC parameters

<table>
<thead>
<tr>
<th>Parameter name or position</th>
<th>Description</th>
<th>Initial value</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 1</td>
<td>This parameter specifies the generic unit name from which the BMC product should dynamically allocate SORTWK data sets. This parameter applies only when the Data Facility Storage Management System (DFSMS) product from IBM is not installed or is not active for temporary DASD work data sets. If DFSMS is active, use the SC parameter.</td>
<td>SYSDA</td>
<td>Use a unit name up to 8 characters.</td>
</tr>
<tr>
<td>Position 2</td>
<td>Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>3</td>
<td>Do not change this value.</td>
</tr>
<tr>
<td>Position 3</td>
<td>This parameter tells the BMC product whether to dynamically allocate SORTWK files.</td>
<td>ON</td>
<td>• ON dynamically allocates SORTWK.</td>
</tr>
<tr>
<td></td>
<td>BMC recommends that you not change this value.</td>
<td></td>
<td>• OFF does not dynamically allocate SORTWK.</td>
</tr>
<tr>
<td>Position 4</td>
<td>Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>ON</td>
<td>Do not change this value.</td>
</tr>
<tr>
<td>Position 5</td>
<td>Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>6000000</td>
<td>Do not change this value.</td>
</tr>
<tr>
<td>Parameter name or position</td>
<td>Description</td>
<td>Initial value</td>
<td>Valid values</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Position 6</td>
<td>Do not change this value. The BMC product does not use this parameter, but the parameter is required for proper assembly of the installation options macro.</td>
<td>3000000</td>
<td>Do not change this value.</td>
</tr>
</tbody>
</table>
| Position 7                | This parameter specifies the DASD type with the smallest track capacity that a dynamically allocated SORTWK data set might encounter at your site. | 3390 | ▪ 3380, track capacity of 47968  
▪ 3390, track capacity of 56664  
▪ 9345, track capacity of 46456 |
| SC                        | This parameter specifies the name of the DFSMS storage class from which to dynamically allocate SORTWK. If DFSMS is active and you do not specify a value for this parameter, the BMC product uses the value from the first DYNALOC parameter.  
**Note:** If your installation has an automatic class selection (ACS) routine, it can override this specification. | Blank | Use any valid DFSMS storage class. |
| RETRY                     | This parameter specifies how you want the BMC product to handle retry attempts for SORTWK dynamic allocation:  
▪ The first subparameter indicates the number of times that you want the BMC product to retry the request.  
▪ The second subparameter indicates the number of minutes to wait between each retry.  
Using this parameter allows you to avoid a capacity-exceeded condition when disk space is not immediately available for a SORTWK dynamic allocation request.  
BMC recommends that you do not change this value because it can affect the elapsed time of your jobs. However, if you currently use SyncSort and rely on the retry function, BMC recommends that you use the same values as your SyncSort RETRY installation parameter. | (0,0) | If you use this parameter, BMC recommends that you specify the same values as your SyncSort RETRY installation parameter. The following values are valid for this parameter:  
▪ 0 through 16 for the first subparameter  
0 indicates that you do not want the BMC product to retry the request.  
▪ 0 through 15 for the second subparameter  
0 indicates that you do not want the BMC product to retry the request. |
DYNALOC installation option
Common utility tables

This chapter describes the contents of the common utility tables, considerations for these tables, and how to maintain them if necessary.

Overview of common utility tables

The BMC common utility tables contain information about the BMC utilities that you generate and submit through a BMC utility product.

Table 30 on page 605 lists the tables that each utility uses and each table’s default name and alias.

Note
CHECK PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS (available only as version 11.2) use synonyms instead of aliases.

Table 30: Common utility tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Default name</th>
<th>Alias</th>
<th>Utilities that use this table</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCDICT</td>
<td>CMN_BMCDICT</td>
<td>BMC_BMCDICT</td>
<td>LOADPLUS, REORG PLUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMCHIST</td>
<td>CMN_BMCHIST</td>
<td>BMC_BMCHIST</td>
<td>CHECK PLUS, NGT Copy, LOADPLUS, NGT Recover, REORG PLUS, UNLOAD PLUS</td>
</tr>
<tr>
<td>Table</td>
<td>Default name</td>
<td>Alias</td>
<td>Utilities that use this table</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| BMCSYNC    | CMN_BMCSYNC      | BMC_BMCSYNC   | ■ CHECK PLUS  
■ NGT Check  
■ NGT Copy  
■ DASD MANAGER PLUS (BMCSTATS)  
■ LOADPLUS  
■ NGT Load  
■ NGT Recover  
■ RECOVERY MANAGER  
■ REORG PLUS  
■ NGT Reorg  
■ UNLOAD PLUS  
■ NGT Unload |
| BMCTRANS   | CMN_BMCTRANS     | BMC_BMCTRANS  | ■ Log Master  
■ RECOVERY MANAGER |
| BMCUTIL    | CMN_BMCUTIL      | BMC_BMCUTIL   | ■ CHECK PLUS  
■ NGT Check  
■ NGT Copy  
■ DASD MANAGER PLUS (BMCSTATS)  
■ LOADPLUS  
■ NGT Load  
■ NGT Recover  
■ RECOVERY MANAGER  
■ REORG PLUS  
■ NGT Reorg  
■ UNLOAD PLUS  
■ NGT Unload |
| BMCXCOPY   | CMN_BMCXCOPY     | BMC_BMCXCOPY  | ■ NGT Copy  
■ Log Master  
■ NGT Recover  
■ RECOVERY MANAGER  
■ REORG PLUS  
■ UNLOAD PLUS  
■ NGT Unload |
Warnings and considerations for common utility tables

This topic describes important information that you need to know when using the common utility tables.

**WARNING**

The following warnings apply to the common utility tables:

- To prevent unpredictable results, do not run any of the following products against the BMC common utility tables or table spaces:
  - LOADPLUS
  - NGT Load
  - REORG PLUS
  - NGT Reorg
  - UNLOAD PLUS
  - NGT Unload

- Because NGT Recover uses BMC tables during the recovery process, you cannot use NGT Recover to recover any BMC table except the BMCHIST table.

- Do not run the RUNSTATS utility against the BMC common utility tables. Doing so can negatively impact utility performance.

- BMC strongly recommends that you use the ISOLATION (UR) bind option and issue SQL COMMIT statements when querying the tables in the BMC database. If objects in the BMC database are restricted for UPDATE, the executing BMC utilities might not be able to complete successfully.

Note the following considerations:

- Some columns in the tables are present for compatibility with specific BMC utilities and are not used by all of the utilities.

- If you have applications that depend on the structure or content of these tables, be aware that these tables are subject to change.

- In general, the utility tables should not require maintenance, with the exception of BMCHIST.

- You should back up the BMC table spaces on a regular basis to enable recoveries. If you use NGT Copy as the copy utility, you must use SHRLEVEL CHANGE for the following spaces:
  - BMCUTIL
  - BMCHIST
  - BMCSYNC
  - BMCXCOPY
Supported versions of the following BMC products support the LOCKROW installation option:

- CHECK PLUS
- LOADPLUS
- REORG PLUS
- UNLOAD PLUS

If the value of the option is YES (which is the default value), the products use MVS enqueues instead of SQL LOCK TABLE statements to serialize updates to the BMCSYNC and BMcutil tables.

The following BMC products always use MVS enqueues for serialization when updating the BMCSYNC and BMcutil tables:

- NGT Copy
- DASD MANAGER PLUS
- NGT Recover

Managing common utility tables

This topic provides basic procedures for working with the common utility tables.

To determine your site’s table names

The names of the common utility tables can be changed during installation.

1. To determine the names that your site uses, perform one of the following actions:

   - Use your utility to run a job with restart parameters of MAINT and MSGLEVEL(1).
     Specifying MSGLEVEL(1) with MAINT prints the names of the BMC tables that your utility uses and identifies the applied maintenance. The utility does not perform any other processing, and the job ends without affecting any utility that is running.

   - Run the following SQL statement, replacing tableName with a BMC common utility table name (listed in “Overview of common utility tables” on page 605):

     ```sql
     SELECT CREATOR, NAME FROM SYSIBM.SYSTABLES
     WHERE TSNAME='tableName';
     ```

   - Get the names from your DB2 system administrator.

To query the tables

1. Run SQL statements similar to the following examples.
Example

This example queries the BMCXCOPY table to access information about the rows in an index space:

```sql
SELECT *
FROM creatorName.CMN_BMCXCOPY
WHERE DBNAME = 'databaseName'
AND IXNAME = 'indexSpaceName'
ORDER BY START_RBA;
```

This example identifies (from the BMCHIST table) the database name, table space name, elapsed time, and when the utility completed:

```sql
SELECT DBNAME, SPNAME, CHAR(ELAPSED, ISO), CHAR(TIME, ISO)
FROM creatorName.CMN_BMCHIST
WHERE UTILID = 'utilityID';
```

To display BMC utility status

1. Use one of the following methods to display the status of BMC utilities:

   - To display the status of all BMC utilities that are executing or awaiting restart for a given table space or index space, use the following SQL statements:

     ```sql
     SELECT *
     FROM creatorName.CMN_BMCUTIL
     WHERE DBNAME = 'databaseName'
     AND SPNAME = 'tableSpaceName'
     SELECT *
     FROM creatorName.CMN_BMCSYNC
     WHERE NAME1 = 'databaseName'
     AND NAME2 = 'spaceName';
     ```

   - If you have a license for the NGT Check, NGT Load, NGT Reorg, NGT Stats, or NGT Unload product, specify NGTDISP BMCUTIL or NGTDISP BMCSYNC to display information about the utilities that are executing or awaiting restart. You can optionally filter this information by utility ID.

     For more information, see the *BMC Next Generation Technology General User Guide*.

To terminate a BMC utility

1. To terminate a BMC utility, perform one of the following actions:

   - To terminate a BMC utility that is executing, use the following SQL statements:

     ```sql
     DELETE FROM creatorName.CMN_BMCUTIL
     WHERE UTILID = 'utilityID';
     DELETE FROM creatorName.CMN_BMCSYNC
     WHERE UTILID = 'utilityID';
     DELETE FROM creatorName.CMN_BMCDICT -- for LOADPLUS and REORG PLUS
     WHERE UTILID = 'utilityID';
     ```

     The utility terminates with return code 8 when the next checkpoint is taken.

   - To clean up a BMC utility that is not executing, run the utility with the correct utility ID and specify the TERM restart parameter.
BMCDICT table

The BMCDICT table stores the compression dictionary during load or reorganization processing.

Table 31 on page 610 describes the contents of the BMCDICT table.

Table 31: Contents of the BMCDICT table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
<tr>
<td>DBNAME</td>
<td>CHAR(8)</td>
<td>Database name</td>
</tr>
<tr>
<td>TSNAME</td>
<td>CHAR(8)</td>
<td>Table space name</td>
</tr>
<tr>
<td>PARTITION</td>
<td>SMALLINT</td>
<td>Partition number For a nonpartitioned table space, the value is 0.</td>
</tr>
<tr>
<td>SEQNO</td>
<td>SMALLINT</td>
<td>Sequence number</td>
</tr>
<tr>
<td>DICTDATA</td>
<td>VARCHAR(4000)</td>
<td>Dictionary data</td>
</tr>
</tbody>
</table>

BMCDICT table considerations

This topic describes important information that you need to know about the BMCDICT table:

- If you are processing a large number of compressed partitions, you might need to increase the size of the BMCDICT table space significantly from the standard size that was allocated during installation. To estimate the allocation, multiply 64 KB by the number of compressed partitions that you are processing concurrently (loading with LOADPLUS or reorganizing with REORG PLUS).

- LOADPLUS inserts rows into the BMCDICT table during the PRELOAD phase and deletes those rows following compression processing in the LOAD phase.

- REORG PLUS inserts rows into the BMCDICT table during the UNLOAD phase and deletes those rows following compression processing in the RELOAD phase.

Maintaining the BMCDICT table

If LOADPLUS or REORG PLUS abends during the time between building the compression dictionary and completing compression, rows might remain in the BMCDICT table.
On rare occasions, you might need to take action to control expansion of the BMCDICT table.

To control expansion of the BMCDICT table

1. Delete any rows in the BMCUTIL table that you know are no longer valid.
   Do not delete any rows for instances of utilities that are awaiting restart.

2. Use the following SQL statement to delete rows from the BMCDICT table:

   ```sql
   DELETE
   FROM creatorName.CMN_BMCDICT
   WHERE UTILID NOT IN
   (SELECT UTILID FROM creatorName.CMN_BMCUTIL);
   ```

   **Note**
   The names of the BMCUTIL and BMCDICT tables might have been changed at your site during installation.

BMCHIST table

The BMCHIST table contains information about completed executions of the BMC utilities for DB2.

The following configuration or installation options control use of the BMCHIST table:

- HISTORY (for NGT Copy, NGT Recover, and UNLOAD PLUS)
- BMCHIST (for REORG PLUS)

If the option value is NO, the utility bypasses any updates to the BMCHIST table. If the value is YES (or the utility does not use a configuration or installation option), the utility inserts rows into the BMCHIST table during the UTILTERM phase.

For NGT Copy, if the value is SUMMARY, the utility inserts only summary information about the NGT Copy execution into the BMCHIST table. This option provides less information than the YES option.

Table 32 on page 611 describes the contents of the BMCHIST table.

**Table 32: Contents of the BMCHIST table**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBNAME</td>
<td>CHAR(8)</td>
<td>Name of the database that contains the table or index space</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SPNAME</td>
<td>CHAR(8)</td>
<td>Name of the table or index space</td>
</tr>
<tr>
<td>UTILNAME</td>
<td>CHAR(8)</td>
<td>Name of the utility: CHECK, COPY, LOAD, RECOVER, REORG, UNLOAD</td>
</tr>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
<tr>
<td>AUTHID</td>
<td>CHAR(8)</td>
<td>User ID that ran the utility</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>Date that the utility completed</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>Time that the utility completed</td>
</tr>
<tr>
<td>ELAPSED</td>
<td>TIME</td>
<td>Elapsed time of the utility</td>
</tr>
<tr>
<td>PARTITION</td>
<td>LONG VARCHAR</td>
<td>ALL, or the partition numbers as specified by the DSNUM option (NGT Copy) or the PART option</td>
</tr>
<tr>
<td>OBJNAME</td>
<td>VARCHAR(27)</td>
<td>Fully qualified object name</td>
</tr>
<tr>
<td>PHASE_1</td>
<td>CHAR(8)</td>
<td>Name of utility phase 1</td>
</tr>
<tr>
<td>ELAPSED_1</td>
<td>TIME</td>
<td>Elapsed time of phase 1</td>
</tr>
<tr>
<td>PHASE_2</td>
<td>CHAR(8)</td>
<td>Name of utility phase 2</td>
</tr>
<tr>
<td>ELAPSED_2</td>
<td>TIME</td>
<td>Elapsed time of phase 2</td>
</tr>
<tr>
<td>PHASE_3</td>
<td>CHAR(8)</td>
<td>Name of utility phase 3</td>
</tr>
<tr>
<td>ELAPSED_3</td>
<td>TIME</td>
<td>Elapsed time of phase 3</td>
</tr>
<tr>
<td>PHASE_4</td>
<td>CHAR(8)</td>
<td>Name of utility phase 4</td>
</tr>
</tbody>
</table>

Note the following conditions:
- This column lists only three-digit partitions (any loaded partitions 1 through 999). Four-digit partitions (any loaded partitions from 1000 through 4096) are not stored in this column. For jobs that load only four-digit partitions, this column is empty.
- If the list of partitions exceeds 1011 bytes, the utility truncates the value that is stored in this column.
- For UNLOAD PLUS, if you specified LOGICAL PART, these partitions are the physical partitions that correspond to the logical partitions that you specified.
BMCHIST table considerations for NGT Copy

NGT Copy uses the BMCHIST table to record completed COPY and COPY IMAGECOPY command executions.

HISTRETN is available as an NGT Copy installation option or as an option on the OPTIONS command. HISTRETN tells NGT Copy the number of days to keep entries in the BMCHIST table.

**WARNING**
If you want to use BMCHIST, allocate adequate space for the table. NGT Copy makes an entry in the table for every copied space. If you are copying a large number of partitions, you might need to increase the size of the BMCHIST table space from the standard size that was allocated during installation.

BMCHIST table considerations for NGT Recover

For each execution of AFRMAIN, NGT Recover writes a single row to the BMCHIST table.

DBNAME, SPNAME, and OBJNAME columns will always be blank.

NGT Recover accumulates elapsed time for each of the following phases using the NGT Recover phase shown:

- **PHASE_1**: LOGSORT
- **PHASE_2**: MERGE (includes RESTORE phase)
- **PHASE_3**: SNAP
- **PHASE_4**: REBUILD (includes UNLOAD phase)
- **PHASE_5**: DB2UTIL (the time spent in DSNUTILB)

The elapsed time for each of the phases is a sum for all objects. The utility elapsed time, ELAPSED, is the duration from the start of the utility to until it finishes. Because NGT Recover multitasks, the sum of the phases might be greater than the total elapsed time of the utility. The elapsed time columns have a limit of 24 hours.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAPSED_4</td>
<td>TIME</td>
<td>Elapsed time of phase 4</td>
</tr>
<tr>
<td>PHASE_5</td>
<td>CHAR(8)</td>
<td>Name of utility phase 5</td>
</tr>
<tr>
<td>ELAPSED_5</td>
<td>TIME</td>
<td>Elapsed time of phase 5</td>
</tr>
</tbody>
</table>

BMCHIST table

Appendix C  Common utility tables 613
Maintaining the BMCHIST table

When a utility completes successfully, it inserts a row into the BMCHIST table. You can control expansion of this table by deleting old rows. If you use REORG PLUS, you can also control inserts into the BMCHIST table.

To delete old rows from the BMCHIST table

1. To delete selected rows from the BMCHIST table based on the date that the utility completed, use the following sample SQL statement:

   ```sql
   DELETE
   FROM creatorName.CMN_BMCHIST
   WHERE DATE < 'yyyy-mm-dd';
   ```

To control inserts into the BMCHIST table (REORG PLUS only)

1. Use the TERMEXIT option to specify a user exit that controls inserts into the BMCHIST table.

BMCSYNC table

The BMCSYNC table contains information about the status of the objects that the currently executing utilities are accessing.

Table 33 on page 615 describes the contents of the BMCSYNC table. The BMCSYNC table synchronizes and controls access to DB2 spaces by concurrently executing BMC utility products. If you have more than one BMC utility installed, all of these utilities should share the same BMCSYNC table.

The following NGT utilities insert rows into BMCSYNC during the BEFOREACC phase and delete rows during the AFTERACC phase:

- NGT Check
- NGT Load
- NGT Reorg
- NGT Stats
- NGT Unload

All other utilities insert rows into BMCSYNC during the UTILINIT phase and delete rows during the UTILTERM phase; while the job executes, the utilities update the table when the status of the object changes.
Table 33: Contents of the BMCSYNC table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(NGT Recover) This column is blank when a RECOVER UNLOADKEYS command creates the row and then a RECOVER BUILDINDEX command reads and deletes the row.</td>
</tr>
<tr>
<td>NAME1</td>
<td>CHAR(8)</td>
<td>Database name or creator name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DASD MANAGER PLUS) This value is the database name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(CHECK PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS) If the value for NAME1 would exceed 8 bytes or the value for NAME2 would exceed 18 bytes, NAME1 contains the DBID for the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload) If the value for NAME1 would exceed 8 bytes, NAME1 contains the OBID in hexadecimal format.</td>
</tr>
<tr>
<td>NAME2</td>
<td>CHAR(18)</td>
<td>Space, table, or index name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(DASD MANAGER PLUS) The BMCSTATS utility always inserts the space name (limited to a maximum of 8 characters).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(CHECK PLUS, LOADPLUS, REORG PLUS, and UNLOAD PLUS) If the value for NAME1 would exceed 8 bytes or the value for NAME2 would exceed 18 bytes, NAME2 contains the table OBID or index ISOBID of the object in hexadecimal format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload) If the value for NAME2 would exceed 18 bytes, NAME2 contains the OBID in hexadecimal format.</td>
</tr>
<tr>
<td>KIND</td>
<td>CHAR(2)</td>
<td>Type of object:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ IP (index partition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ IX (index)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ TB (table)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ TP (table space partition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ TS (table space)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ DD, DW, D1, D2 (dynamic work file allocation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ CI (copy information)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RD (restart data set block)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ LK (limit key)</td>
</tr>
<tr>
<td>PARTITION</td>
<td>SMALLINT</td>
<td>Physical partition number:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Null or 0 for a single data set nonpartitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Data set number for a multi-data-set, nonpartitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Partition number for a partitioned space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(all products except NGT Recover and RECOVERY MANAGER) The value is null or 0 for any nonpartitioned space.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| BMCID       | SMALLINT  | Internal identifier of the object  
DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg,  
NGT Stats, and NGT Unload do not use this column. |
| UTILNAME    | CHAR(8)   | Name of the executing utility:  
- CHECK  
- CHECKIX  
- COPY  
- STATS  
- LOAD  
- NGTSTATS  
- REBUILD  
- RECOVER  
- REORG  
- UNLOAD |
| SHRLEVEL    | CHAR(1)   | Degree to which utilities can share this object:  
- Blank means that no status is requested, and any other utility  
can obtain any status.  
- S allows sharing among any number of SHRLEVEL S  
utilities.  
- X indicates that exclusive control is required. No other utility  
can run with SHRLEVEL X.  
For more information, see “Shared access levels of BMC  
utilities” on page 619. |
| STATUS      | CHAR(1)   | Status of the utility or object:  
- Blank (indicates no processing has been done)  
- C (for CHECK PLUS, indicates checked)  
- L (for LOADPLUS, indicates loaded)  
- U (for UNLOAD PLUS, indicates unloaded)  
- R (for REORG PLUS, indicates reloaded)  
DASD MANAGER PLUS does not use this column. |
| XCOUNT      | INTEGER   | Number of rows or keys processed in the current phase  
DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg,  
NGT Stats, and NGT Unload do not use this column. |
| DDNAME      | CHAR(8)   | Check, load, unload, or work ddname  
DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg,  
NGT Stats, and NGT Unload do not use this column. |
| BLOCKS      | INTEGER   | Number of blocks for the check, load, unload, or work data set  
DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg,  
NGT Stats, and NGT Unload do not use this column. |
**BMCSYNC table considerations**

This topic contains important information that you need to know about the BMCSYNC table:

- By default, DASD MANAGER PLUS uses the BMCSYNC table to synchronize access to DB2 spaces. However, if you want to turn this feature off, you may do so by specifying No for the BMCSYNC installation option. If you specify No for this option, DASD MANAGER PLUS does not use the BMCSYNC table and the product bypasses BMCUTIL table access, UTILID enqueue logic, and object name enqueue logic used for BMC utility concurrency control. Turning this feature off can lead to VSAM data set access failures in BMCSTATS or other utilities due to utility conflicts that are no longer detected.

- You might need to increase the size of the BMCSYNC table space from the standard size that was allocated during installation when any of the following conditions exists:
  - You are processing a large number of partitions.
  - Estimate this allocation based on the following factors:
    - Number of utilities that you are executing concurrently
    - Number of partitions that you are processing concurrently
    - Number of files that you are allocating dynamically
—You are loading a partition-by-growth table space.
   Estimate this allocation based on the following factors:
   —Number of utilities that you are running concurrently
   —Value of MAXPARTITIONS
   —Number of files that you are allocating dynamically

—You are loading or unloading XML data and the XML table space is partition-
   by-growth.
   Estimate this allocation based on the following factors:
   —Number of utilities that you are executing concurrently
   —Number of XML columns that you are loading or unloading
   —Value of MAXPARTITIONS (a minimum of 256 partitions in this case)
   —Number of files that you are allocating dynamically

—You are loading or unloading LOB data.
   Estimate this allocation based on the following factors:
   —Number of utilities that you are executing concurrently
   —Number of LOB columns that you are loading or unloading
   —Number of partitions in the base table space
   —Number of files that you are allocating dynamically

Maintaining the BMCSYNC table

When a utility abends, rows might remain in the BMCSYNC table. On rare
occasions, you might need to take action to control expansion of the BMCSYNC
table.

To control expansion of the BMCSYNC table

1 Use one of the following methods to delete rows in the BMCSYNC table:
   ■ Use the TERM restart parameter on the EXEC statement to delete rows from
     both the BMCUTIL and BMCSYNC tables. Do not delete any rows for instances
     of utilities that are awaiting restart.

   ■ Delete invalid rows from the BMCUTIL table. Do not delete any rows for
     instances of utilities that are awaiting restart.

Then use the following SQL statement to delete rows from the BMCSYNC
table:

```
DELETE FROM creatorName.CMN_BMCSYNC
WHERE UTILID NOT IN
  (SELECT UTILID FROM creatorName.CMN_BMCUTIL);
```
Cleaning up RECOVER UNLOADKEYS entries

Successful completion of a RECOVER UNLOADKEYS job leaves rows in BMCSYNC with blank utility IDs for table space partitions and indexes related to the unloaded keys. The table space rows prevent other BMC utilities from obtaining exclusive control of the table space.

To clean up RECOVER UNLOADKEYS entries

1. Use one of the following methods to remove the invalid BMCSYNC rows:

   - Run a RECOVER BUILDINDEX job.
   - Run a job that uses the following statement for the table space and each index:

     ```sql
     DELETE FROM creatorName.CMN_BMCSYNC
     WHERE UTILID=''
     AND NAME1='databaseName'
     AND NAME2='spaceName'
     AND UTILNAME='RECOVER';
     ```

Shared access levels of BMC utilities

BMC utility jobs register DB2 objects in the BMCSYNC table.

The registering utility assigns a sharing level to each registered object. The sharing level controls access to that object from other BMC utilities. For partitioned DB2 spaces, registration is performed at the partition level.

Note

All BMC utility products use the BMCUTIL table to control the use of utility IDs, which identify executions of BMC utilities. Each BMC utility product must have a unique ID for restart purposes. This unique ID is stored in the BMCUTIL table. For more information about this table, see “BMCUTIL table” on page 623.

The BMCSYNC table allows multiple BMC utilities (or multiple instances of a utility) to operate concurrently on different partitions of a DB2 space if no nonpartitioning indexes are involved. In addition, some BMC utilities can operate concurrently on the same object or partition. For information about which products can operate...
concurrently, see the following table. For additional serialization and concurrency
issues for each utility, see that utility's reference manual.

The "Access level" column in the following table refers to the value of the
SHRLEVEL column name in the BMCSYNC table ("BMCSYNC table" on page 614).
The level can be one of the following values:

- S indicates shared access. Any other utility that registers with shared access (S)
can run against the object.

- X indicates exclusive access. No other utility can run against the object.

- A blank value indicates that no status is requested and any other utility can run
against the object.

Table 34: Shared access levels of BMC utilities

<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK PLUS</td>
<td>S</td>
<td>None</td>
</tr>
</tbody>
</table>
| DASD MANAGER PLUS     | S            | ■ If BMCSTATS is processing multiple objects and encounters an object that is held by another utility, the BMCSTATS job issues a warning. The warning identifies the object and the utility that is using it. BMCSTATS continues processing the next object.  
■ If BMCSTATS is processing an object and another utility requires exclusive control of that object, the other utility stops execution at initialization time. |
<p>| (BMCSTATS)            |              |                                                                                        |
| LOADPLUS              | X            | If you specify PART, LOADPLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions while running this job. |
| NGT Check             | S            | None                                                                                    |
| NGT Copy              | S or blank   | If you specify COPY IMAGECOPY, NGT Copy registers the object with no access status (blank). Otherwise, NGT Copy registers the object with shared access (S). |
| NGT Load              | X            | If you are loading specific partitions, NGT Load registers only the specified partitions with exclusive access (X). |</p>
<table>
<thead>
<tr>
<th>Product</th>
<th>Access level</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGT Recover</td>
<td>X, S, or blank</td>
<td>NGT Recover registers an object with shared access (S) under the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ The table space for an index is registered with shared access if the index is being rebuilt and its table space is not recovered in the same job.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ A table space partition is registered with shared access if the keys for that partition are unloaded with a RECOVER UNLOADKEYS operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGT Recover registers an object with no access status (blank) if you specify the following commands or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ The ACCUM command</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ OUTCOPY ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ INDEP OUTSPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGT Recover registers the object with exclusive access (X) in all other cases.</td>
</tr>
<tr>
<td>NGT Reorg</td>
<td>X</td>
<td>If you are loading specific partitions, NGT Reorg registers only the specified partitions with exclusive access (X).</td>
</tr>
<tr>
<td>NGT Stats</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>NGT Unload</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>RECOVERY MANAGER</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>REORG PLUS</td>
<td>X</td>
<td>If you specify PART, REORG PLUS registers only the specified partitions with exclusive access (X). If no nonpartitioned indexes exist on the table space, you can run other utilities on different partitions while running this job.</td>
</tr>
<tr>
<td>UNLOAD PLUS</td>
<td>S</td>
<td>None</td>
</tr>
</tbody>
</table>
BMCTRANS table

The BMCTRANS table contains information that RECOVERY MANAGER and Log Master use for transaction recovery.

Table 35 on page 622 describes the contents of the BMCTRANS table. The table contains one row for each execution of Log Master (that is, one row for each log scan performed).

Table 35: Contents of the BMCTRANS table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERID</td>
<td>CHAR(8) NOT NULL</td>
<td>Transaction creator</td>
</tr>
<tr>
<td>TRANID</td>
<td>VARCHAR(18) NOT NULL</td>
<td>Transaction ID</td>
</tr>
<tr>
<td>STARTTIME</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Transaction start time</td>
</tr>
<tr>
<td>PITRBA</td>
<td>CHAR(6) NOT NULL FORBIT DATA</td>
<td>RBA for point-in-time recovery</td>
</tr>
<tr>
<td>OUTDSNAME</td>
<td>VARCHAR(35) NOT NULL</td>
<td>Output data set prefix for SQL statements or the logical log</td>
</tr>
<tr>
<td>STATE</td>
<td>SMALLINT NOT NULL</td>
<td>Level of recovery analysis performed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 (only UNDO analysis has been performed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 through 9999 (UNDO and PIT analysis have been performed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Greater than 10000 (UNDO, PIT, and REDO analysis have been performed)</td>
</tr>
<tr>
<td>PITTIME</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Timestamp for the PIT RBA</td>
</tr>
<tr>
<td>SEQNO</td>
<td>SMALLINT NOT NULL</td>
<td>Sequence number of the filter text</td>
</tr>
<tr>
<td>PITWKEST</td>
<td>FLOAT NOT NULL</td>
<td>Work estimate</td>
</tr>
<tr>
<td>FILTERLINE</td>
<td>VARCHAR(1040) NOT NULL</td>
<td>Text of the filter (may span more than one row)</td>
</tr>
<tr>
<td>UNDONUMROWSUPD</td>
<td>FLOAT</td>
<td>Number of unique rows (RIDs) that are selected by the filter of the log scan</td>
</tr>
<tr>
<td>UNDOSUBSEQUPDROWS</td>
<td>FLOAT</td>
<td>Total number of anomaly log records relating to one of the rows (RIDs) selected by the log scan</td>
</tr>
<tr>
<td>UNDOLOGRECROWS</td>
<td>FLOAT</td>
<td>Number of unique rows (RIDs) that are affected by an anomaly log record</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UNDOJOBSTATUS</td>
<td>SMALLINT</td>
<td>Code indicating the status of an UNDO log scan:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 (no action taken)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 (Log Master execution started)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 2 (Log Master execution completed successfully with return code 0,4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 3 (Log Master execution completed unsuccessfully with return code 8,12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 4 (Log Master execution abnormally ended)</td>
</tr>
<tr>
<td>REDOJOBSTATUS</td>
<td>SMALLINT</td>
<td>Code indicating the status of a REDO log scan:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 (no action taken)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 (Log Master execution started)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 2 (Log Master execution completed successfully with return code 0,4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 3 (Log Master execution completed unsuccessfully with return code 8,12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 4 (Log Master execution abnormally ended)</td>
</tr>
<tr>
<td>ENDTIME</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Transaction end time</td>
</tr>
<tr>
<td>ACTION</td>
<td>SMALLINT</td>
<td>Code indicating what recovery, if any, has been performed on the transaction</td>
</tr>
</tbody>
</table>

**BMCUTIL table**

The BMCUTIL table contains information about utilities that are currently running or started.

*Table 36 on page 624* describes the contents of the BMCUTIL table. The utilities use the table to control the use of utility IDs. Each BMC utility must have a unique ID for...
restart purposes. If you have more than one BMC utility installed, all of these utilities should share the same BMCUTIL table.

The utilities insert rows into the BMCUTIL table during the UTILINIT phase and update the table as the job status changes. The utilities delete rows from the BMCUTIL table during the UTILTERM phase.

Table 36: Contents of the BMCUTIL table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILID</td>
<td>CHAR(16)</td>
<td>Utility identifier</td>
</tr>
<tr>
<td>STATUS</td>
<td>CHAR(1)</td>
<td>Execution status of the utility: A (active, not executing command), I (initializing), P (pausing or pause-stopped), S (stopped), T (terminating), X (executing command)</td>
</tr>
<tr>
<td>UTILNAME</td>
<td>CHAR(8)</td>
<td>Name of the executing utility: CHECK, COPY, STATS, LOAD, NGT Load, RECOVER, REORG, UNLOAD</td>
</tr>
<tr>
<td>PHASE</td>
<td>CHAR(8)</td>
<td>Current phase of the utility. NGT Copy does not use this column. (NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload) The value for this column is always UTILINIT.</td>
</tr>
<tr>
<td>USERID</td>
<td>CHAR(8)</td>
<td>User ID executing the utility</td>
</tr>
<tr>
<td>SSID</td>
<td>CHAR(4)</td>
<td>DB2 subsystem where the utility is running</td>
</tr>
<tr>
<td><strong>Column name</strong></td>
<td><strong>Data type</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>RESTART</strong></td>
<td>CHAR(1)</td>
<td>Restart option:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (not restart)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ P (RESTART(PHASE))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (RESTART)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASD MANAGER PLUS does not use this column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload)</em> The value for this column is always N.</td>
</tr>
<tr>
<td><strong>NOTEID</strong></td>
<td>CHAR(8)</td>
<td>TSO user ID to be notified DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
<tr>
<td><strong>DBNAME</strong></td>
<td>CHAR(8)</td>
<td><em>(NGT Recover and REORG PLUS)</em> Name of the database containing the table or index space for which the last checkpoint was taken This value can be blank.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The other utilities do not use this column.</td>
</tr>
<tr>
<td><strong>SPNAME</strong></td>
<td>CHAR(8)</td>
<td><em>(NGT Recover and REORG PLUS)</em> Name of the table or index space for which the last checkpoint was taken This value can be blank.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The other utilities do not use this column.</td>
</tr>
<tr>
<td><strong>SPSTATUS</strong></td>
<td>CHAR(5)</td>
<td><em>(REORG PLUS)</em> Space status before the utility stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The other utilities do not use this column.</td>
</tr>
<tr>
<td><strong>COMMANDNO</strong></td>
<td>SMALLINT</td>
<td><em>(NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload)</em> Always 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For all other utilities, this column is always 0.</td>
</tr>
<tr>
<td><strong>COMMAND</strong></td>
<td>VARCHAR(256)</td>
<td>First 256 characters of the utility command text NGT Recover, DASD MANAGER PLUS, NGT Copy, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
<tr>
<td><strong>STATE</strong></td>
<td>LONG VARCHAR</td>
<td>Utility state and sync information DASD MANAGER PLUS, NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
<tr>
<td><strong>START_TIMESTAMP</strong></td>
<td>TIMESTAMP</td>
<td>Starting timestamp of the utility NGT Check, NGT Load, NGT Reorg, NGT Stats, and NGT Unload do not use this column.</td>
</tr>
</tbody>
</table>
Maintaining the BMCUTIL table

When a utility abends, rows might remain in the BMCUTIL table.

On rare occasions, you might need to take action to control expansion of the BMCUTIL table.

To control expansion of the BMCUTIL table

1 Use one of the following methods to delete rows from the BMCUTIL table:

- Use the TERM restart parameter on the EXEC statement to delete rows from both the BMCUTIL and BMCSYNC tables. Do not delete any rows for instances of utilities that are awaiting restart.

- Delete invalid rows in the BMCUTIL table. Do not delete any rows for instances of utilities that are awaiting restart.

Then use the following SQL statement to delete rows from the BMCSYNC table:

```sql
DELETE FROM creatorName.CMN_BMCSYNC
WHERE UTILID NOT IN
(SELECT UTILID FROM creatorName.CMN_BMCUTIL);
```

Note
The names of the BMCUTIL and BMCSYNC tables might have been changed at your site during installation.

BMCXCOPY table

The BMC utilities use the BMCXCOPY table to track registered copies. Table 37 on page 627 describes the contents of the BMCXCOPY table, which contains information about the following types of registered copies:

- Indexes that NGT Copy has copied:
  - COPY NO index copies
  - DSNUM n index (nonpartitioned) copies
  - Incremental index copies
  - Index copies that are made at data set level

- Instant Snapshots made by NGT Copy that are not registered as Flash Copies in SYSCOPY with the BMC EXTENDED BUFFER MANAGER (XBM) product or
BMC SNAPSHOT UPGRADE FEATURE (SUF) technology, and any standard copies made in association with the Instant Snapshot

- Online consistent copies
- Cabinet copies
- Encrypted copies

The BMCXCOPY table functions like SYSIBM.SYSCOPY except that IXNAME replaces TSNAME in BMCXCOPY. You must control authorization and access to users for BMCXCOPY through standard DB2 authorization.

If you have more than one BMC utility installed, all of these utilities should share the same BMCXCOPY table.

Table 37: Contents of the BMCXCOPY table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBNAME</td>
<td>CHAR(8)</td>
<td>Name of the database</td>
</tr>
<tr>
<td>IXNAME</td>
<td>CHAR(8)</td>
<td>Name of the index space or table space for Instant Snapshots and associated copies</td>
</tr>
<tr>
<td>DSNUM</td>
<td>INTEGER</td>
<td>Data set number within the index or table space</td>
</tr>
<tr>
<td>ICTYPE</td>
<td>CHAR(1)</td>
<td>Operation type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- F (COPY FULL YES; for NGT Copy, online consistent copies)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- I (COPY FULL NO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- W (REORG LOG NO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- B (REBUILD INDEX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- P (POINT-IN-TIME RECOVERY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- C (for NGT Copy version 7.3 and earlier, online consistent copies)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- s (used by NGT Copy to track system pages)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- m (indicates that the table space was exported by the NGT Copy EXPORT command or migrated by the NGT Recover IMPORT command)</td>
</tr>
<tr>
<td>ICDATE</td>
<td>CHAR(6)</td>
<td>Date of the entry (yymmd)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| START_RBA   | VARCHAR(10)  | The relative byte location of a point in the DB2 recovery log<br>The indicated point as follows:  
  ■ For ICTYPE F, the starting point for all updates since the image copy was taken  
  ■ For COPY_TYPE O, the minimum of the consistent point and the oldest inflight URID  
  ■ (RECOVERY MANAGER) For ICTYPE C, the consistent log point for the copy  
    — RBA for non-data-sharing systems  
    — LRSN for data sharing systems |
| FILESEQNO   | INTEGER      | Tape file sequence number of the copy                                                                                                    |
| DEVTYPE     | CHAR(8)      | Type of device on which the copy resides                                                                                                 |
| IBMREQD     | CHAR(1)      | Whether the row came from the basic machine-readable material (MRM) tape:  
  ■ N (NO)  
  ■ Y (YES) |
| DSNAME      | CHAR(44)     | Name of the data set  
  If STYPE V, DSNAME is the name of the VSAM data component.                                                                                               |
| ICTIME      | CHAR(6)      | Time at which this row was inserted (hhmmss)  
  The insertion takes place after the completion of the operation that the row represents.                                                                 |
| SHRLEVEL    | CHAR(1)      | SHRLEVEL parameter on COPY if ICTYPE F:  
  ■ C (change)  
  ■ R (reference) |
| DSVOLSER    | VARCHAR(1784)| Volume serial numbers of the data set  
  Commas separate items in a list of 6-byte numbers. This column is blank if the data set is cataloged.                                                     |
| TIMESTAMP   | TIMESTAMP    | Date and time when the row was inserted  
  This column contains the date and time that are recorded in ICDATE and ICTIME. The use of TIMESTAMP over ICDATE and ICTIME is recommended, because later DB2 releases might not support the latter two columns. |
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| ICBACKUP      | CHAR(2)  | Type of image copy contained in the data set:  
|               |           | ■ LB (data set contains local backup data)  
|               |           | ■ RP (data set contains recovery system main data)  
|               |           | ■ RB (data set contains recovery system backup data)  
|               |           | ■ Blank (data set contains local system main data or is not one of multiple copies) |
| ICUNIT        | CHAR(1)  | Media on which the image copy data set is stored:  
|               |           | ■ D (DASD)  
|               |           | ■ T (tape)  
|               |           | ■ Blank (medium is neither tape nor DASD) |
| STYPE         | CHAR(1)  | Type of copy:  
|               |           | ■ Blank (for ICTYPE=F)  
|               |           | ■ V (Instant Snapshot or a VSAM data set)  
|               |           | ■ e (encrypted copy) |
| PIT_RBA       | VARCHAR(10) | Point-in-time recovery:  
|               |           | ■ X'000000000000' (for ICTYPE=F)  
|               |           | ■ Consistent point (for COPY_TYPE=O) |
| GROUP_MEMBER  | CHAR(8)  | Data-sharing group member (the name of the SSID where the copy was made)  
|               |           | This column is blank if you are not using data sharing. |
| OTYPE         | CHAR(1)  | Type of object:  
|               |           | ■ T (table)  
|               |           | ■ I (index)  
|               |           | ■ i (compressed index) |
| LOWDSNUM      | INTEGER  | Not used |
| HIGHDSNUM     | INTEGER  | Not used |
| COPIYPAGESF   | FLOAT(53) | Number of pages written to the copy data set |
| NPAGESF       | FLOAT(53) | High-used RBA divided by the page size |
| CPAGESF       | FLOAT(53) | Total number of changed pages |
| JOBNAME       | CHAR(8)  | Job name |
| AUTHID        | CHAR(8)  | Authorization ID |
| OLDEST_VERSION | SMALLINT | When ICTYPE= B, F, I, S, W, or X, the version number of the oldest format of data for an object  
<p>|               |           | For other values of ICTYPE, the value is -1. |
| LOGICAL_PART  | INTEGER  | Logical partition number |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGGED</td>
<td>CHAR(1)</td>
<td>Logging attribute of the table space:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (logged)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (not logged)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (row inserted prior to DB2 version 9) For a non-LOB table space or index space, blank indicates that the logging attribute is logged.</td>
</tr>
<tr>
<td>TTYPE</td>
<td>CHAR(8)</td>
<td>Row format for the table space or partition:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RRF (reordered row format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ BRF (basic row format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RBA/LRSN format for the space or partition:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ B (basic 6-byte format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ E (extended 10-byte format)</td>
</tr>
<tr>
<td>INSTANCE</td>
<td>SMALLINT</td>
<td>Instance number of the current base objects (table and index) The default value is 1.</td>
</tr>
<tr>
<td>RELCREATED</td>
<td>CHAR(1)</td>
<td>DB2 release that created the object If the release is earlier than Version 9, the value is blank.</td>
</tr>
<tr>
<td>COPY_TYPE</td>
<td>CHAR(1)</td>
<td>Type of copy:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ C (cabinet copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ O (online consistent copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ X (export copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ I (import copy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (default value)</td>
</tr>
<tr>
<td>NOTE_VALUE</td>
<td>CHAR(4)</td>
<td>Encoded value that quickly locates data for a specific space in a cabinet copy The default value is blank.</td>
</tr>
<tr>
<td>NOTE_TYPE</td>
<td>CHAR(1)</td>
<td>Type of NOTE (issued by NGT Copy):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ A (ABS - tape)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ R (REL - disk)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ F (frame)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (default value)</td>
</tr>
<tr>
<td>OCC_COPY_RBA</td>
<td>VARCHAR(10)</td>
<td>Original START_RBA of an online consistent copy The default value is blank.</td>
</tr>
<tr>
<td>OCC_LOCKRULE</td>
<td>CHAR(1)</td>
<td>Locking rule for a table space (not used for indexes):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ A (for page level)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ R (for row level)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Blank (default value)</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
OCC_SPACE_ALTERED | CHAR(1) | Whether the space was altered:  
- Y (altered)  
- N (not altered)  
- Blank (default value)
CAB_BLOCKS | INTEGER | Total number of frames written for a cabinet copy
EXPSSID | VARCHAR(8) | Source location SSID of the migration file (valid with COPY_TYPE = I)
EXPSSLRSN | VARCHAR(10) | Indicates the SYNC AUTO point on the source (valid with COPY_TYPE = I and COPY_TYPE = X)
EXPTLRSN | VARCHAR(10) | Indicates the SYNC AUTO point on the target (valid with COPY_TYPE = I)

### Maintaining the BMCXCOPY table

Periodically, you should review BMCXCOPY and delete old rows to control its expansion.

#### To control expansion of the BMCXCOPY table

1. To delete all rows from the BMCXCOPY table that are older than 30 days, run an SQL DELETE statement, using the following statement as an example:

```sql
DELETE
FROM creatorName.CMN_BMCXCOPY
WHERE DAYS(CURRENT_TIMESTAMP) - DAYS(TIMESTAMP) > 30;
```
BMC Common DB2 repository tables

The BMC common DB2 repository is made up of several DB2 tables.

Naming conventions

The BMC common DB2 repository tables follow a naming convention. The following table provides the synonyms and local table names.

Note
The local table names might be different at your site, based on options selected during product installation.

<table>
<thead>
<tr>
<th>Synonym</th>
<th>Local table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMCSCC_OBJSETS</td>
<td>BMCUTIL.CMN_OS</td>
</tr>
<tr>
<td>BMCSCC_OBJSET_DEF</td>
<td>BMCUTIL.CMN_OS_DEF</td>
</tr>
<tr>
<td>BMCSCC_OBJSET_SQL</td>
<td>BMCUTIL.CMN_OS_SQL</td>
</tr>
<tr>
<td>BMCSCC_GRPOPTS</td>
<td>BMCUTIL.CMN_OS_OPTS</td>
</tr>
<tr>
<td>BMCSCC_PRODREG</td>
<td>BMCUTIL.CMN_OS_PREG</td>
</tr>
<tr>
<td>BMCSCC_GROUPAUTH</td>
<td>BMCUTIL.CMN_OS_GAUTH</td>
</tr>
</tbody>
</table>

OBJSETS table

The following table describes the contents of the OBJSETS table. This table describes and provides information about object sets. This table contains one row for each object set defined in the repository.
## OBJSET_DEF table

The following table describes the contents of the OBJSET_DEF table. This table contains one row for each object set definition specification defined for an object set.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSNAME</td>
<td>VARCHAR(27) NOT NULL</td>
<td>Name of object set</td>
</tr>
<tr>
<td>SEQNO</td>
<td>SMALLINT NOT NULL</td>
<td>Sequence number of definition</td>
</tr>
<tr>
<td>CREATE_TSMP</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Timestamp of object set creation</td>
</tr>
<tr>
<td>CREATE_UID</td>
<td>CHAR(8) NOT NULL</td>
<td>AUTHID of creator of the object set</td>
</tr>
<tr>
<td>UPDATE_TSMP</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Timestamp of last maintenance activity</td>
</tr>
<tr>
<td>UPDATE_UID</td>
<td>CHAR(8) NOT NULL</td>
<td>AUTHID of last updater of the object set</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(60) NOT NULL</td>
<td>Description of the object set</td>
</tr>
<tr>
<td>PRODUCT_ID</td>
<td>CHAR(3) NOT NULL</td>
<td>Creating product ID</td>
</tr>
<tr>
<td>TYPE</td>
<td>CHAR(2) NOT NULL</td>
<td>Product group type</td>
</tr>
<tr>
<td>NUMBER_OBJECTS</td>
<td>INTEGER NOT NULL WITH DEFAULT</td>
<td>Number of objects from last open</td>
</tr>
<tr>
<td>CHECKSUM</td>
<td>SMALLINT NOT NULL</td>
<td>Verification value from API updates</td>
</tr>
<tr>
<td>OSNAME_DELIMITED</td>
<td>CHAR(1) NOT NULL WITH DEFAULT 'N'</td>
<td>For use with delimited names</td>
</tr>
</tbody>
</table>

### Column Description:

- **OSNAME**: Name of the object set.
- **CREATE_TSMP**: Timestamp of object set creation.
- **CREATE_UID**: AUTHID of creator of the object set.
- **UPDATE_TSMP**: Timestamp of last maintenance activity.
- **UPDATE_UID**: AUTHID of last updater of the object set.
- **DESCRIPTION**: Description of the object set.
- **PRODUCT_ID**: Creating product ID.
- **TYPE**: Product group type.
- **NUMBER_OBJECTS**: Number of objects from last open.
- **CHECKSUM**: Verification value from API updates.
- **OSNAME_DELIMITED**: For use with delimited names. Will be 'Y' if the related column is a delimited name (entered with double quotes around it) when entered.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCEXC_IND</td>
<td>CHAR(1) NOT NULL</td>
<td>Include or exclude indicator (+, -)</td>
</tr>
<tr>
<td>PATTERN_TYPE</td>
<td>CHAR(2) NOT NULL</td>
<td>Pattern for include or exclude:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ TS (table space name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ IX (index name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ TB (table name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ IS (index space name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ PL (plan name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ PG (package name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ SG (stogroup name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ OS (object set name pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ SQ (dynamic SQL pattern)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ RP (repository plan)</td>
</tr>
<tr>
<td>INC_IX</td>
<td>CHAR(1) NOT NULL</td>
<td>Include related indexes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (Yes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (No)</td>
</tr>
<tr>
<td>INC_RI</td>
<td>CHAR(1) NOT NULL</td>
<td>Include RI objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (Yes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (No)</td>
</tr>
<tr>
<td>INC_LOBS</td>
<td>CHAR(1) NOT NULL</td>
<td>Include LOB objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (Yes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (No)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>INC_XML</td>
<td>CHAR(1) NOT NULL</td>
<td>Include XML objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (Yes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (No)</td>
</tr>
<tr>
<td>INC_CLONES</td>
<td>CHAR(1) NOT NULL</td>
<td>Include clones only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (Yes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (No)</td>
</tr>
<tr>
<td>BY_PART</td>
<td>CHAR(1) NOT NULL</td>
<td>Expand objects by partition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Y (Yes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (No)</td>
</tr>
<tr>
<td>PART_BEG</td>
<td>SMALLINT NOT NULL</td>
<td>Beginning partition number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0-4096)</td>
</tr>
<tr>
<td>PART_END</td>
<td>SMALLINT NOT NULL</td>
<td>Ending partition number (0-4096)</td>
</tr>
<tr>
<td>REF_SEQ_NBR</td>
<td>SMALLINT NOT NULL</td>
<td>For future use</td>
</tr>
<tr>
<td>DESC</td>
<td>VARCHAR(60) NOT NULL</td>
<td>Description of the specification</td>
</tr>
<tr>
<td>OBJ_QUAL1</td>
<td>VARCHAR(128) NOT NULL</td>
<td>Object qualifier 1</td>
</tr>
<tr>
<td>OBJ_QUAL2</td>
<td>VARCHAR(128) NOT NULL</td>
<td>Object qualifier 2</td>
</tr>
<tr>
<td>OBJ_QUAL3</td>
<td>VARCHAR(128) NOT NULL</td>
<td>Object qualifier 3</td>
</tr>
<tr>
<td>UNIQUALS</td>
<td>CHAR(1) NOT NULL</td>
<td>UNICODE indicator</td>
</tr>
<tr>
<td>UPDATE_UID</td>
<td>CHAR(8) NOT NULL</td>
<td>ID of last updater of object set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>definitions</td>
</tr>
<tr>
<td>UPDATE_TSMP</td>
<td>TIMESTAMP NOT NULL WITH</td>
<td>Timestamp of last maintenance</td>
</tr>
<tr>
<td></td>
<td>DEFAULT</td>
<td>activity</td>
</tr>
<tr>
<td>PACKAGE_VERSION</td>
<td>SMALLINT NOT NULL</td>
<td>Package version</td>
</tr>
<tr>
<td>INC_HISTORY</td>
<td>CHAR(1) NOT NULL WITH</td>
<td>Include related history objects</td>
</tr>
<tr>
<td></td>
<td>DEFAULT 'N'</td>
<td>■ Y (Yes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ N (No)</td>
</tr>
</tbody>
</table>
## OBJSET_SQL table

The following table describes the contents of the OBJSET_SQL table. This table contains one row for each object set specification in dynamic SQL (type SQ).

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSNAME</td>
<td>VARCHAR(27) NOT NULL</td>
<td>Name of the object set</td>
</tr>
<tr>
<td>SPEC_SEQNO</td>
<td>SMALLINT NOT NULL</td>
<td>Sequence number from OBJSET_DEF table</td>
</tr>
<tr>
<td>SEQNO</td>
<td>SMALLINT NOT NULL</td>
<td>Sequence number to order multiple SQL entries</td>
</tr>
<tr>
<td>TEXT</td>
<td>VARCHAR(72) NOT NULL</td>
<td>Line of SQL text</td>
</tr>
</tbody>
</table>

## GRPOPTS table

The following table describes the contents of the GRPOPTS table. This table contains one row for each option defined to either a defined group, or a subsystem level option.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSNAME</td>
<td>VARCHAR(27) NOT NULL</td>
<td>Name of object set</td>
</tr>
</tbody>
</table>
### OPTION_TYPE

**Data type**: CHAR(10) NOT NULL  
**Description**: Option type  
- Backup—ARMOPTBKUP  
- Recover—ARMOPTRCVR  
These are the option types currently used by RECOVERY MANAGER. The option type is defined by the product, so this list is product-dependent.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTION_TYPE</td>
<td>CHAR(10) NOT NULL</td>
<td>Option type</td>
</tr>
<tr>
<td>OPTION</td>
<td>VARCHAR(200) NOT NULL</td>
<td>Option name</td>
</tr>
<tr>
<td>OPT_VALUE</td>
<td>VARCHAR(200) NOT NULL</td>
<td>Value for named option</td>
</tr>
</tbody>
</table>

### PRODREG table

The following table describes the contents of the PRODREG table. There should be one entry for each product and version that is registered.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT_ID</td>
<td>CHAR(3) NOT NULL</td>
<td>Product ID</td>
</tr>
<tr>
<td>PLAN_NAME</td>
<td>VARCHAR(24) NOT NULL</td>
<td>Plan name</td>
</tr>
<tr>
<td>PRODUCT_VERSION</td>
<td>CHAR(4) NOT NULL</td>
<td>Product version</td>
</tr>
</tbody>
</table>

### GROUPAUTH table

The following table describes the contents of the GROUPAUTH table. This table optionally contains one row for each authority granted on a group. No rows exist if no authority has been granted.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSNAME</td>
<td>VARCHAR(27) NOT NULL</td>
<td>Name of object set</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>CHAR(8) NOT NULL</td>
<td>AUTHID to whom authorization was granted</td>
</tr>
<tr>
<td>TYPE</td>
<td>CHAR(1) NOT NULL</td>
<td>Type of authorization granted</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>CHAR(8) NOT NULL</td>
<td>Grantor of authorization</td>
</tr>
<tr>
<td>DATE_GRANTED</td>
<td>TIMESTAMP NOT NULL WITH DEFAULT</td>
<td>Timestamp of when authorization was granted</td>
</tr>
</tbody>
</table>
NGT Recover syntax diagrams

For quick reference, this appendix provides the syntax diagrams for the NGT Recover commands without any option descriptions. Cross-references to the option descriptions are included in the diagrams.

Alphabetical listing of NGT Recover options

NGT Recover options are listed in the following table, alphabetized by NGT Recover command, and within the command by option name.

**IMPORT command options**

- ANALYZE
- AUX
- CHANGE ... FROM...TO
- CLONE
- DEFINE
- DSNUM
- EXCLUDE
  - For OBJECTSET specification
  - For USING specification
- INDEP OUTSPACE
- INDEXES
- MODEL
- NOCOPYPEND
- NOSCRATCH
- OBJECTSET
- OUTCOPY
- OUTCOPYDDN
- REBUILD
- RECOVERYDDN
- REDEFINE
For information about these options, see “IMPORT” on page 290.

**LOGSCAN command options**

- **AUX**
- **BACKOUT**
- **CLONE**
- **DSNUM**
- **FROMLOGPOINT**
  - LASTARCHQ
  - LASTCOPY
  - LASTQUIESCE
  - LASTSHUTDOWN
  - X’logPoint'
- **FROMRBA**
  - LASTARCHQ
  - LASTCOPY
  - LASTQUIESCE
  - LASTSHUTDOWN
  - X’logPoint'
- **INDEX**
  - For single index specification
  - For multiple index specification
- **INDEXES (or INDEX)**
- **INDEXSPACE**
  - For single index specification
  - For multiple index specification
- **LOGAPPLY ONLY**
- **LOGONLY**
- **OBJECTSET**
- **TABLESPACE**
- **TOLOGPOINT**
— LASTARCHQ
— LASTCOMMONQ
— LASTQUIESCE (relativeGenerationNumber)
— LASTSHUTDOWN
— X'logPoint'

■ TORBA
— LASTARCHQ
— LASTCOMMONQ
— LASTQUIESCE (relativeGenerationNumber)
— LASTSHUTDOWN
— X'logPoint'

For information about these options, see “LOGSCAN” on page 253.

**MIGRATE command options**

■ ANALYZE
■ CHANGE ... FROM...TO
■ CLONE
■ DSNUM
■ EXCLUDE
  — For OBJECTSET specification
  — For USING specification
■ FROMLOGPOINT
  — LASTARCHQ
  — LASTCOPY
  — LASTQUIESCE
  — LASTSHUTDOWN
  — X'logPoint'
■ FROMRBA
  — LASTARCHQ
  — LASTCOPY
  — LASTQUIESCE
  — LASTSHUTDOWN
  — X'logPoint'
■ INDEP OUTSPACE
■ LOGAPPLY ONLY
■ LOGONLY
■ LOGSORT
■ MODEL
- NUMREC
  - ABS
  - AVGRECSZ
  - CALC
  - EST
  - NOEST
- OBJECTSET
- RESET
- SORTDEVT
- SORTNUM
- TABLESPACE
- TOCOPY LASTCOPY
- TOLOCATION
- TOLOGPOINT
  - LASTARCHQ
  - LASTCOMMONQ
  - LASTQUIESCE (relativeGenerationNumber)
  - LASTSHUTDOWN
  - LOGMARK logMarkName (logMarkGeneration)
  - X’logPoint'
- TORBA
  - LASTARCHQ
  - LASTCOMMONQ
  - LASTQUIESCE (relativeGenerationNumber)
  - LASTSHUTDOWN
  - LOGMARK logMarkName (logMarkGeneration)
  - X’logPoint'
- USING

For information about these options, see “MIGRATE” on page 270.

**OPTIONS command options**

- ANALYZE
- AUTOSIZE
- AUX
- BACKOUT
- CHECKINT
- DATAMVR
- DEFINE
Alphabetical listing of NGT Recover options

- EARLYCAT
- EARLYRECALL
- INDEXLOG
- IXRECP
- KEYSORT
- KSORTSHARE
- LOGPOINT X'\logPoint' or LOGPOINT LOGMARK logMarkName (logMarkGeneration)
- LOGSORT
- MAXDRIVES
- MAXKSORT
- MAXLOGS
- MAXLSORT
- NOEARLYCAT
- NOEARLYRECALL
- NOSYSLGRNG
- NUMREC
  - ABS
  - AVGRECSZ
  - CALC
  - EST
  - NOEST
- ON ERROR ANY CONTINUE
- OUTCOPY
  - ASCODED
  - BYPART
- REBUILDSCOPE
- RECOVERSCOPE
- RECOVERYPOINT
- RESINV
- RESOURCE SELECTION
  - ACCUMS
  - COPIES
  - LOGS
- RICHK
- SIMULATE YES
- SMCORE
- SNAP
- SORTDEVT
  - For KEYSORT
— For LOGSORT
  ■ SORTDIAG
  ■ SORTDYN
  ■ SORTNUM
    — For KEYSORT
    — For LOGSORT
  ■ STOGROUP... USEORDER
  ■ TIMESTAMP
  ■ TRTCH
  ■ URIDDDN
  ■ USEACCUM
  ■ USEHDROBIDS
  ■ WTOR
  ■ XBMID
  ■ ZIIP

For information about these options, see “OPTIONS command” on page 103.

OUTPUT command options

■ CATLG
■ DATACLAS
■ DSNNAME
■ EATTR
■ EXPDT
■ MAXPRIM
■ MGMTCLAS
■ MODELDDB
■ RETPD
■ SPACE
■ STACK
■ STORCLAS
■ TRTCH
■ UNIT
■ UNITCNT
■ VOLUMES

For information about these options, see “OUTPUT command” on page 142.
REBUILD INDEX and SIMRBLD INDEX command options

- ANALYZE
- AUX
- CLONE
- DEFINE
- INDEP INTABLESPACE
- INDEP OUTSPACE
- MODEL
  - For INDEP INTABLESPACE
  - For INDEP OUTSPACE
- NOSCRATCH
- NUMREC
  - ABS
  - CALC
  - EST
  - NOEST
- NOWORKDDN
- OBJECTSET
- PART
- REDEFINE
- REUSE
- SCOPE
  - ALL
  - PENDING
  - STATUS\((status1, status2, \ldots)\)
- SORTDEVT
- SORTNUM
- TABLESPACE
- WORKDDN

For information about these options, see “REBUILD INDEX” on page 215.

RECOVER BUILINDEX and SIMRCVR BUILINDEX command options

- ANALYZE
- CLONE
- INDEP OUTSPACE
- MODEL
Alphabetical listing of NGT Recover options

- NOSCRA
- REDEFINE
- REUSE
- SKEY
- SKEYDDN
- TABLESPACE

For information about these options, see “RECOVER BUILDINDEX” on page 246.

RECOVER INDEX, RECOVER INDEXSPACE, RECOVER TABLESPACE, SIMRCVR INDEX, SIMRCVR INDEXSPACE, and SIMRCVR TABLESPACE command options

- ANALYZE
- AUX
- BACKOUT
- CLONE
- DBID
- DEFINE
- DROPRECOVERY
- DSNAM
- DSNUM
- — For INDEX
- — For INDEXSPACE
- — For TABLESPACE
- DSSIZE
- ENCRYPTED
- FROMLOGPOINT
  — LASTARCHQ
  — LASTCOPY
  — LASTQUIESCE
  — LASTSHUTDOWN
  — X'logPoint'
- FROMRBA
  — LASTARCHQ
  — LASTCOPY
  — LASTQUIESCE
  — LASTSHUTDOWN
  — X'logPoint'
- INCOPY FULL
- INCR
- INDEP OUTSPACE
- INDEVT
- INDEX
  - For single index specification
  - For multiple index specification
- INDEXES (or INDEX)
- INDEXSPACE
  - For single index specification
  - For multiple index specification
- INLINE
- INLOG LOGPOINT
- INLOG RBA
- INSEQNO
- INVOLUME
- LASTCOPY (relativeGenerationNumber)
- LOCALSITE
- LOGAPPLY ONLY
- LOGONLY
- LOGPOINT
- LOGSORT
- MODEL
- NOCOPYPEND
- NOSCRATCH
- NOWORKDDN
- NUMREC
  - ABS
  - AVGRECSZ
  - CALC
  - EST
  - NOEST
- OBID
- OBIDXLAT
- OBJECTSET
- OUTCOPY
- OUTCOPYDDN
- OUTCOPYDSN
- PIECESIZE
- PSID
- RBA
- RECOVERYDDN
- RECOVERYDSN
- RECOVERYSITE
- REDEFINE
- REGISTER
- RESET
- RESETRSTS
- REUSE
- RICHK
- SCOPE
  - ALL
  - PENDING
  - STATUS(status1, status2, …)
  - UPDATED
- SHRLEVEL
- SNAPSHOT
- SORTDEVT
- SORTNUM
- SPACE
- TABLESPACE
  - For table space specification
  - For multiple index specification
- TOCOPY
- TOLOGPOINT
  - LASTARCHQ
  - LASTCOMMONQ
  - LASTQUIESCE (relativeGenerationNumber)
  - LASTSHUTDOWN
  - LOGMARK logMarkName (logMarkGeneration)
  - X'logPoint'
- TORBA
  - LASTARCHQ
  - LASTCOMMONQ
  - LASTQUIESCE (relativeGenerationNumber)
  - LASTSHUTDOWN
  - LOGMARK logMarkName (logMarkGeneration)
  - X'logPoint'
- TOSEQNO
- TOVOLUME
- TRANSFORM
- UPDATE VERSIONS

For information about these options, see “RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE commands” on page 151.

**RECOVER UNLOADKEYS and SIMRCVR UNLOADKEYS command options**

- ANALYZE
- CLONE
- INDEPENDENT TABLESPACE
- MODEL
- NUMREC
  - ABS
  - CALC
  - EST
  - NOEST
- PART
- SKEY
- SKEDDN
- SORTDEVT
- SORTNUM
- TABLESPACE

For information about these options, see “RECOVER UNLOADKEYS” on page 237.

**OPTIONS command syntax diagram**

The following syntax diagrams define the options that can be used with the OPTIONS command.
Figure 190: OPTIONS syntax diagram

650  BMC Next Generation Technology Recover for DB2 for z/OS Reference Manual
Figure 191: OPTIONS syntax diagram, continued

1 Valid with R/CHANGE ACCUM only
2 Requires a Recovery Management for DB2 solution password
Figure 192: OPTIONS syntax diagram, continued

OPTIONS command syntax diagram

1 Requires a Recovery Management for DB2 solution password
The following syntax diagrams define the options that can be used with the OUTPUT command.
Figure 196: OUTPUT syntax

```
OUTPUT name DSNAMES dataSetName
```

Common options

Disk options

Tape options

Figure 197: OUTPUT syntax--Common options

```
CATLG YES NO
DATACLASS name
MGMTCLASS name
MODELDCB dataSetName
STORCLASS name
UNIT SYSALDA name
UNITCOUNT 25 integer
VOLCOUNTER 25 integer
```

Figure 198: OUTPUT syntax--Disk options

```
EATTR NO YES NONE
MAXPRIM 0 integer
SPACE (primary, secondary)
CYL TRK
VOLUMES (volume1, volume2, ...)
```
The following syntax diagrams define the options that can be used with the RECOVER TABLESPACE, RECOVER INDEX, RECOVER INDEXSPACE commands.
Figure 200: RECOVER TABLESPACE, RECOVER INDEX, and RECOVER INDEXSPACE syntax diagram

- **RECOVER**
  - **SIMRCVR**
    - Table space specification
    - Single index specification
    - Multiple index specification
    - OBJECTSET specification
  - **FROMRBA** or **FROMLOGPOINT** specification
  - **INCOPY** specification
- **INDEPENDENT**
  - **OUTSPACE** specification
- **LOGSORT** specification
- **Non-registered copy or INLOG** specification
- **OBIDXLAT** specification
- **OUTCOPY** specification
- **OUTCOPYDDN** specification
- **RECOVERYDDN** specification
- **TOCOPY** specification
- **TORBA** or **TOLOGPOINT** recovery specification

- **ANALYZE**
  - **YES**
  - **NO**
  - **ONLY**
- **CLONE**
- **DEFINE**
  - **YES**
  - **NO**
- **LOCALSITE**
- **RECOVERYSITE**
- **LOGAPPLY ONLY**
- **LOGONLY**

- **REDEFINE**
  - **YES**
  - **NOSCRATCH**
  - **NO**
- **REUSE**

- **UPDATE VERSIONS**

- **SCOPE** specification

1 Not valid with BACKOUT
Figure 201: RECOVER syntax—Table space specification

Table space specification

- TABLESPACE
  - DATABASE
    - DATABASENAME
  - OBJECTSET specification
  - INDEXES
    - NO
    - YES
  - INDEX
    - HISTORY
    - ARCHIVE
  - DSCNUM
    - ALL
      - integer
    - AUX
      - NO
      - YES
    - begin: end
  - DROPRECOVERY
  - RICHK
    - NO
    - YES
  - TRANSFORM
    - Non-registered copy or INLOG specification
    - OUTCOPY specification
  - FROMRBA or FROMLOGPOINT specification
  - OUTCOPY specification

1 Not valid with TRANSFORM
2 Not valid with BACKOUT
Figure 202: RECOVER syntax—Single index specification

Single index specification

INDEX (creatorID, indexName)

OBJECTSET specification

INDEXSPACE (databaseName, indexSpaceName)

DSNDB04

DSNUM

ALL

integer

begin : end

NOWORKDDN

OBIDXLAT specification

NOWORKDDN

Non-registered copy or INLOG specification

FROMRBA or FROMLOGPOINT specification

OUTCOPY specification

1 Valid with INDEXLOG AUTO
Figure 203: RECOVER syntax—Multiple index specification

---

1. Allowed with multiple recoveries only if dynamic outcopy allocation (OUTPUT) syntax is used.
2. Valid with INDEXLOG AUTO.
Figure 204: RECOVER syntax—OBJECTSET specification

**OBJECTSET specification**

- OBJECTSET
  - creator.name

- EXCLUDE
  - dbname.
  - qualifier.ix
  - DSNUM
  - ALL
  - integer
  - begin:end

- OBIDXLAT
  - RESET

Figure 205: RECOVER syntax—INDEPENDENT OUTSPACE specification

**INDEPENDENT OUTSPACE**

- INDEP OUTSPACE
  - MODEL
    - Vcat.MDBDBC.dbname.objectName.0001.znnn
  - MODEL
    - userName.getDataSet

^ Not valid with BACKOUT

Figure 206: RECOVER syntax—TORBA or TOLOGPOINT specification

**TORBA or TOLOGPOINT specification**

- TORBA
- TOLOGPOINT
- LASTQUIESCE
  - (relativeGenerationNumber)
  - BACKOUT
  - YES
- LASTCOMMONQ
- LASTARCHQ
- LASTSHUTDOWN
- LOGMARK logMarkName
  - (logMarkGeneration)
- ‘x’logPoint
Figure 207: RECOVER syntax—TOCOPY specification

**TOCOPY specification**

- TOCOPY
- LASTCOPY
  - (relativeGenerationNumber)
  - dataSetName
- TOVOLUME
- CATALOG
  - volumeSerialNumber
- TOSEQNO integer

*Not valid with OBJECTSET*

Figure 208: RECOVER syntax—LOGSORT specification

**LOGSORT specification**

- LOGSORT
  - SORTDEVT deviceType
  - SORTNUM integer
  - NUMREC
    - CALC
    - NOEST
    - EST integer
    - ABS integer
      - AVGRECSZ integer
Figure 209: RECOVER syntax—OBIDXLAT specification

**OBIDXLAT specification**

1. **OBIDXLAT**
   - **RESET**
     - **DBID**
       - `X'hexSourceID'`
       - `decimalSourceID`
       - `X'hexTargetID'`
       - `decimalTargetID`
     - **PSID**
       - `X'hexSourceID'`
       - `decimalSourceID`
       - `X'hexTargetID'`
       - `decimalTargetID`
   - **OBID**
     - `X'hexSourceID'`
     - `decimalSourceID`
     - `X'hexTargetID'`
     - `decimalTargetID`

1 Not valid with BACKOUT
Figure 210: RECOVER syntax—Non-registered copy or INLOG specification

Non-registered copy or INLOG specification

INCOPY — FULL
    | INCOPY specification
    | INCR
    | NOCOPYPEND
    | X'logPoint'
    | INLOG RBA
    | INLOG LOGPOINT

1 Not valid with index specification
2 Not valid with BACKOUT

Figure 211: RECOVER syntax—FROMRBA or FROMLOGPOINT specification

FROMRBA or FROMLOGPOINT specification

FROMRBA
FROMLOGPOINT
LASTQUIESCE
LASTCOPY
LASTARCHQ
LASTSHUTDOWN
X'logPoint'

Figure 212: RECOVER syntax—OUTCOPY specification

OUTCOPY specification

OUTCOPY NO YES
    | REGISTER
    | ALL
    | RESETRTS
    | ONLY
    | REGISTER
    | ALL
    | RESETRTS
    | ddname

1 Not valid with BACKOUT or Instant Snapshot copies
Figure 213: RECOVER syntax—OUTCOPY specification, continued

OUTCOPY specification, continued

OUTCOPYDDN specification

TAG tagName

SPACE (pri, sec)

CYL

RECOVERYDDN specification

TRK
Figure 214: RECOVER syntax—INCOPY specification

INCOPY specification

DSNAME dataSetName
  MODEL dataSetName
    DSSIZE (integerG)¹
    PIECESIZE (...)¹
      integerK
      integerM
      integerG

RBA
LOGPOINT

X’logPoint’

SHRLEVEL
REFERENCE
  CHANGE

ENCRIPTED
TIMESTAMP
  timestamp

COMPRESSED ²

INVOLUME
CATALOG
  volumeSerialNumber
  INDEV deviceType

INSEQNO integer

¹ Valid only with TRANSFORM
² Not valid with table space specification
Figure 215: RECOVER syntax—OUTCOPYDDN specification

OUTCOPYDDN specification

```
OUTCOPYDDN ( DDName1, DDNamePrefix1, outputDescriptor1, DDName2, DDNamePrefix2, outputDescriptor2 )
```

1 These default values apply to table spaces with fewer than 100 partitions. If the table space has 100 partitions or more, the corresponding default values are BMCCY and BMCCZ.
2 These values are the default values used by the OUTPUT command.
3 These values are used to specify the OUTPUT command, which will control the allocation of the copy.

Figure 216: RECOVER syntax—RECOVERYDDN specification

RECOVERYDDN specification

```
RECOVERYDDN ( DDName3, DDNamePrefix3, outputDescriptor3, DDName4, DDNamePrefix4, outputDescriptor4 )
```

1 These default values apply to table spaces with fewer than 100 partitions. If the table space has 100 partitions or more, the corresponding default values are BMCRCY and BMCRCZ.
2 These values are the default values used by the OUTPUT command.
3 These values are used to specify the OUTPUT command, which will control the allocation of the copy.
The following syntax diagrams define the options that can be used with the REBUILD INDEX command.

Figure 217: REBUILD INDEX syntax diagram

1 Is synonymous with REBUILD when INDEXLOG=NO
2 Is synonymous with SIMRBLD when INDEXLOG=NO
Figure 218: REBUILD INDEX syntax diagram, continued

Figure 219: REBUILD INDEX syntax diagram--OBJECTSET specification

OBJECTSET specification

OBJECTSET  creator.name

EXCLUDE ( dbname.ts, qualifier.ix, DNUM ALL integer begin : end )
The following syntax diagrams define the options that can be used with the RECOVER UNLOADKEYS command.
Figure 221: RECOVER UNLOADKEYS syntax diagram

RECOVER UNLOADKEYS (indexName 
(PART 1 integer begin : end 
(--- ALL ---)) 

TABLESPACE DSNDB04. 
databaseName. 

INDEP INTABLESPACE 

CLONE SKEYDDN 

NUMREC 

SKEY DDName 

SORTDEVT deviceType 

SORTNUM integer 

MODEL vc. BMDDBC.databaseName.getTableSpaceName.10001.znn 

MODEL userNamedDataSet 

ANALYZE YES 

NO 

ONLY

1 You cannot specify PART in both places.
RECOVER BUILDINDEX command syntax diagram

The following syntax diagrams define the options that can be used with the RECOVER BUILDINDEX command.

Figure 222: RECOVER BUILDINDEX syntax diagram

LOGSCAN command syntax diagram

The following syntax diagrams define the options that can be used with the LOGSCAN command.
**Figure 223: LOGSCAN syntax diagram**

```
LOGSCAN
  ▼
  Table space specification
    ▼
    Single index specification
      ▼
      Multiple index specification
    ▼
    OBJECTSET — creator.name
      ▼
      TORBA or TOLOGPOINT recovery specification
        ▼
        LOGAPPLY ONLY
          ▼
          LOGONLY
```

1 Not valid with BACKOUT, which is part of the TORBA to TOLOGPOINT specification

**Figure 224: LOGSCAN syntax—Table space specification**

```
TABLESPACE
  ▼
  tableSpaceName
    ▼
    DSNDB04
      ▼
      databaseName.
    ▼
    OBJECTSET — creator.name
      ▼
      INDEXES
        ▼
        INDEX
          ▼
          NO
          ▼
          YES
    ▼
    DNUM
      ▼
      ALL
        ▼
        integer
          ▼
          begin : end
      ▼
      AUX
        ▼
        NO
        ▼
        YES
          ▼
          HISTORY
          ▼
          ARCHIVE
    ▼
    FROMRBA
    ▼
    FROMLOGPOINT
    ▼
    LASTQUIESCE
    ▼
    LASTCOPY
    ▼
    LASTARCHQ
    ▼
    LASTSHUTDOWN
    ▼
    X'logPoint'
```
Figure 225: LOGSCAN syntax—Single index specification

```
INDEX
  ( creatorID, indexName )
  INDEXSPACE ( databaseName, indexSpaceName )
  DSNDB04

FROMRBA
FROMLOGPOINT
  LASTQUIESCE
  LASTCOPY
  LASTARCHQ
  LASTSHUTDOWN
  X‘logPoint’
```

LOGSCAN command syntax diagram
Figure 226: LOGSCAN syntax—Multiple index specification

Multiple index specification

Figure 227: LOGSCAN syntax—TORBA or TOLOGPOINT specification

TORBA or TOLOGPOINT specification
MIGRATE command syntax diagram

The following syntax diagrams define the options that can be used with the MIGRATE command.

Figure 228: MIGRATE syntax

1 Requires a valid password as follows: Recovery Management or Database Administration
Figure 229: MIGRATE syntax--USING specification

**USING specification**

- USING \( \text{dataSelName} \)
  - CHANGE TABLE FROM \( \text{creator1.tableName1} \) TO \( \text{creator2.tableName2} \)
  - CHANGE INDEX FROM \( \text{creator1.indexName1} \) TO \( \text{creator2.indexName2} \)
  - INDEXES
  - RECOVER
  - NO
  - EXCLUDE \( \{ \text{dbname.ts, qualifier.ix} \} \)
  - DSNUM
  - ALL
    - integer
    - begin : end
  - RESET

Figure 230: MIGRATE syntax--FROMRBA or FROMLOGPOINT specification

**FROMRBA or FROMLOGPOINT specification**

- FROMRBA
- FROMLOGPOINT
- LASTQUIESCE
- LASTCOPY
- LASTARCHQ
- LASTSHUTDOWN
- \( X'logPoint' \)
Figure 231: MIGRATE syntax--OBJECTSET specification

**OBJECTSET specification**

- OBJECTSET
- creator.name
- EXCLUDE
- (dbname.ts)
- qualifier.ix

Figure 232: MIGRATE syntax--INDEPENDENT OUTSPACE specification

**INDEPENDENT OUTSPACE specification**

- INDEPENDENT OUTSPACE
- MODEL
- vcat.BMCDBC.databaseName.objectName.I0001.znnn
- userNamedDataSet

Figure 233: MIGRATE syntax--TORBA or TOLOGPOINT specification

**TORBA or TOLOGPOINT specification**

- TORBA
- TOLOGPOINT
- LASTQUIESCE
- (relativeGenerationNumber)
- LASTCOMMONQ
- LASTARCHQ
- LASTSHUTDOWN
- LOGMARK logMarkName
- (logMarkGeneration)
- X'logPoint'
The following syntax diagrams define the options that can be used with the IMPORT command.
Figure 235: IMPORT syntax

1 Required a valid password as follows: Recovery Management or Database Administration
2 Not valid with BACKOUT
Figure 236: IMPORT syntax—USING specification

**USING specification**

```
USING dataSetName

| CHANGE TABLE FROM creator1.tableName1 TO creator2.tableName2 |
| CHANGE INDEX FROM creator1.indexName1 TO creator2.indexName2 |

EXCLUDE (  
  dbname.ts
  qualifier.ix
  ALL
  integer
  begin : end
)

SYNC
  AUTO
  REPLACE

INDEXES
  AUTO
  RECOVER
  REBUILD
  NO

STATS
  NO
  YES
```

Figure 237: IMPORT syntax—OBJECTSET specification

**OBJECTSET specification**

```
OBJECTSET creator.name

EXCLUDE (  
  dbname.ts
  qualifier.ix
)
```
Figure 238: IMPORT syntax—INDEPENDENT OUTSPACE specification

INDEPENDENT OUTSPACE specification

INDEP OUTSPACE

MODEL

vcat.BMCDBC.databaseName.objectName.10001.znnn

MODEL

userNamedDataSet

Figure 239: IMPORT syntax—OUTCOPY specification

OUTCOPY specification

OUTCOPY

NO

YES

OUTCOPYDDN
specification

RECOVERYDDN
specification
Index

@x03DOPT 573
&ATTACH symbolic variable 145, 495
&DAY symbolic variable 145, 496
&DATE symbolic variable 145, 495
&DB symbolic variable 145, 495
&HOUR symbolic variable 145, 496
&ICTYPE symbolic variable 145, 496
&INST symbolic variable 145, 497
&JDATE symbolic variable 145, 495
&JOBNAME symbolic variable 145, 495
&JDAY symbolic variable 145, 496
&JDATE symbolic variable 145, 495
&LDATE symbolic variable 145, 497
&PART symbolic variable 145, 495
&PART4 variable 496
&PART5 variable 497
&SECOND symbolic variable 145, 496
&SEQ symbolic variable 145, 496
&SP symbolic variable 145, 495
&SSID symbolic variable 145, 495
&STEPNAM symbolic variable 145, 495
&TIME symbolic variable 145, 495
&TS symbolic variable 145, 495
&type symbolic variable 145, 496
&UID symbolic variable 145, 495
&UNIQ (or &UQ) symbolic variable 145
&UNIQ (or &UQ) variable 497
&USERID symbolic variable 145, 495
&UTIL symbolic variable 145, 496
&VCAT symbolic variable 145, 495
&YEAR symbolic variable 145, 495
$C30DOPT 573

A
above-the-bar storage 66
access, shared 614
ACCUM command 422
ACHKP status 57
added partitions 503
adding comments 87
adding objects for restart 335
ADRDBSS utility 538
Advanced Encryption Standard (AES) 506
advantages of LOGSORT strategy 460
advantages of sorting the log 459
AES (Advanced Encryption Standard) 506
AFR$OPTS installation options module 573
AFRDBG 319
AFRDUMP 319
AFRERR 319
AFRGDG data set 319, 493, 569
AFRMAIN execution module 335
AFROSUM 319
AFRPLAN 319
AFRPLAN data set 373
AFRPRINT 319
AFRPRnnn 319
AFRSTMT 319
AFRSTMT data set 373
AFRSTMT data set 373
AFRSTMT data set 373
AFRSTMT data set 373
AFRSTMT data set 373
AFRSTMT data set 373
AFRSTMT data set 373
AFRTIME 319
AFRTRACE 319
ALL
SCOPE option 201
allocating copy data sets 491
allocating output data sets 83, 145
allocation of EAV 148, 582
allocation of extended format sequential data set 148, 582
alphabetical listing of options 92, 639
ALTER for DB2 525, 526, 531, 535, 538, 543
AMSCAT option 578
ANALYZE ONLY option 373
ANALYZE option 209, 289, 307
   ANALYZE ONLY 51
   description 130, 232, 245, 251
   using to preview recovery resources and plans 51
ANSI Data Encryption Algorithm (DEA) 506
APF authorization 70, 573
archive log 451
ARCHIVE LOG MODE (QUIESCE) 550, 558, 569
ARCHIVE option 300
archive quiesce points
   migrating from 279
   migrating to 284
   recovering from 180
   recovering to 204
   scanning from 258
   scanning to 266
ASCII data support 473
audit controls 34
authority
   APF 70
   DATAACCESS 68
   DBADM 68
   DBCTRL 68
   DISPLAYDB 68
   IMAGCOPY 68
   installation SYSADM 68
   RACF 69
   RECOVERDB 68
   STARTDB 68
   STOPDB 68
   SYSADM 68
   SYSCTRL 68
   system 69
   system DBADM 68
   to access DB2 subsystem 68
   to run NGT Recover 68
   to run NGT RECOVER 68
   to stop the database 68
authorizations
   mechanisms, description 68
   automatic fallback 465
   automatic sizing of output files 138
   AUTOSIZE option 138, 578
   AUX option 168, 175, 223, 258, 263, 298
   installation option description 578
auxiliary table spaces 56
AVGRECSZ option 44, 110
B
backing up BMC tables 607
BACKOUT AUTO option 47
BACKOUT AUTO option, description 122
BACKOUT option
   and hash spaces 468
   and indexes defined using an expression 468
   and LOB spaces 468
   and LOBs 468
   and NOT LOGGED spaces 468
   and XML spaces 468
   description 122, 266
   example 401
   for TOLOGPOINT 204, 266
   for TORBA 204, 266
   installation option description 580
   recovery without image copies 45
   restrictions 122, 164, 193, 204, 266, 275, 468
backout recovery 47
backup copies, making 193
benefits of NGT Recover 33
bind qualifier 580
BINDQUALIFIER installation option 580
BMC tables 633
BMC utilities
   displaying status 608
   running concurrently 619
   terminating 608
BMCDICT table
   considerations 610
   contents 610
   maintaining 610
BMCHIST installation option, BMCHIST table 611
BMCHIST table
   and the HISTORY installation option 584
   backing up 607
   contents 611
   maintenance 614
   NGT Copy considerations 613
   NGT Recover considerations 613
   querying 608
BMCSORT installation option 601
BMCSYNC table
backing up 607
clean up after job termination 340
cleaning up NGT Recover UNLOADKEYS 619
concurrency 75
considerations 617
contents 614
LOB data considerations 617
maintaining 618
running utilities concurrently 619
utilid deleted 314
XML data considerations 617
BMCTRANS table 622
BMCUTIL table
backing up 607
clean up after job termination 340
concurrency 75
contents 623
maintaining 626
utilid deleted 314
BMCXCOPY table
backing up 607
contents 626
maintaining 631
querying 608
registrations 54
role in index backups 58
build phase 358
BUILDINDEX option
description 248
rebuilding nonpartitioned indexes 50
building a nonpartitioned index 402
building NGT Recover jobs 311
described 487
dynamic sizing 138
example 422
grouping objects 488, 490
CHANGE MANAGER for DB2 525, 526, 531, 535, 538, 543, 544
changed limit keys 504
CHECKINT option
command option description 112
effects on restart 339
installation option description 581
checkpoint override parameter
CHECKPT(NO) 316
CHECKPT(PHASE) 316
controlling overhead 316
effects on restart 339
checkpoints, specifying the time between 112
CHECKPT option
checkpoint override parameter 316
effects on restart 339
installation option description 581
CHECKPT(NO) 316
CHECKPT(PHASE) 316
cleaning up
after a job termination 340
after an UNLOADKEYS job 341
BMCUTIL and BMCSYNC 340
using the TERM parameter 314
CLONE option 210, 227, 242, 250, 265, 288, 306, 514
cloned objects
CLONE option 514
data sets 514
limitations for NGT Recover support 514
recovering 514
commands, alphabetical listing 92, 639
comments coded in statements 87
common DB2 repository tables 633
common repository
naming conventions 633
compatibility
BMC Software products 40
DB2 resources 40
compressed indexes 517
COMPRESSED option 182
compressing data sets, when stacking to tape 150
compression
BMCDICT table 610
concurrent operation

C

CA-ACF2 security product 68
CA-Top Secret security product 68
cabinet copies
overview 60
registration 60, 61
strategies for use 484
CATALOG MANAGER for DB2 525, 526, 531, 535, 543
catalog, ICF 69
CATLG option 146
CHANGE ... FROM ...TO option 295
CHANGE ... FROM...TO option 276
change accumulation file
of UNLOADKEYS and BUILDINDEX 77
recovering multiple partitions 451
recovery of table space and indexes 458
with IBM DB2 utilities 74
with other BMC utilities for DB2 75
with other NGT RECOVER executions 76
consistent point, obtaining 558, 569
controlling output 319
conventions, documentation 16
copies
migrating from the last 279
recovering from the last 180
scanning from the last 258
COPY command option
EXCLUDE 220
COPY command options
EXCLUDE 177
copy data set ddnames 198, 199, 214
copy migration 520
copy registration
BMCXCOPY table 626
copy registration, SHRLEVEL 567
copying cabinet copies 60
CPUMIPS installation option 581
creating
consistent database for query and review 544
consistent set of copies at a point in time 569
consistent set of copies at a point in time,
OUTCOPY ONLY JCL 569
copies from log and prior copies 567, 569
DB2 objects 526, 531, 538, 544, 550
image copy without reading a space 571
migration images 558
creating copies from recovery resources 55
cumulative incremental image copies 78

D
data compression, for tape data sets 150
data definitions, obtaining 526, 531, 535, 538, 544,
550, 558
Data Encryption Standard (DES) 506
data migration 541
data sets

AFRDBG 319
AFRDUMP 319
AFRERR 319
AFRGDG 319
AFROSUM 319
AFRPLAN 319
AFRPRINT 319
AFRPRnnn 319
AFRSTMNT 319
AFRSUMRY 319
AFRTIME 319
AFRTRACE 319
compression, for tape data sets 150
dynamic allocation 83, 145, 319
for log 323
for output copies 198, 199
for REBUILD INDEX 326
for RECOVER BUILDINDEX 328
for RECOVER INDEX 323
for RECOVER INDEXSPACE 323
for RECOVER TABLESPACE 323
for RECOVER UNLOADKEYS 327
for sorts 329
input image copies 323
interim for index keys 50
L001WKnn 329, 333
log sort work 323
LxxxWKnn 329, 333
optional 319
output image copies 323
reallocating 212, 308
required 319
reusing 213, 234, 253, 310
S001WKnn 326, 329
SKEYDDN option 327, 328
STOGROUP-defined 212, 233, 252, 308
STOGROUP-defined, required authority 69
SxxxWKnn 326, 329
symbolic variables 185, 282, 303
SYSERR 319
SYSIN 319
SYSOUnnn 319, 326
SYSPICK 319
SYSPRINT 319
SYSSCAN 319
SYSSDUMP 319
tape stacking considerations 403
VCAT-defined 212, 233, 252, 308
VSAM 212, 233, 308
WORKDDN option 326
data sharing
considerations 500
  group attach name 314
DATAACCESS authority 68
DATACLAS installation option 581
DATACLAS option 146
DATAMVR option 129, 582
date/time values
  Daylight Savings Time adjustments 141, 594
  TZRULE option 141, 594
Daylight Savings Time adjustments 594
DB2
  authorizations needed for NGT RECOVER 68
  commands issued by NGT RECOVER 71
  group attach name in JCL 314
  object status 317
  RACF ID 69
  subsystem ID (SSID) in JCL 314
  subsystem identifier (SSID) 314
DB2 objects, creating 526, 531, 538, 544, 550
DB2 REPAIR VERSIONS 513
DB2 versioning, steps to handle 513
DBADM authority 68
DBCTRL authority 68
DBID 526, 531, 544, 550, 558
DD name construction
  for nonpartitioned spaces 323
  for partitioned spaces 323
DD statements
  common to all executions 319
  for REBUILD INDEX 326
  for RECOVER BUILDINDEX 328
  for RECOVER INDEX 323
  for RECOVER INDEXSPACE 323
  for RECOVER TABLESPACE 323
  for RECOVER UNLOADKEYS 327
  in JCL 319
  input image copies 323
L001WKnn 329, 333
log data sets 323
LxxxWKnn 329, 333
output image copies 323
S001WKnn 326, 329
SKEYDDN option 327, 328
specifying work data sets for index keys 463
STEPLIB 319
SxxxWKnn 326, 329
SYSUT1 326
tape mounts, stacked tapes 323
WORKDDN option 326

DDNAME 402
DEA (Data Encryption Algorithm 506
default option See installation options 573
DEFINE option
  description of installation option 582
DES (Data Encryption Standard 506
DFDSS utility 538
dictionaries, compression
  BMCDICT table 610
directing recovery output 405
DISKIORATE installation option 582
DISPLAYDB authority 68
displaying job status 340
displaying status of BMC utilities 608
DocID 516
documentation
  summary of changes 17
documentation information 15
DOPT See installation options 573
drives, maximum 587
drop recovery
  dropped table from a simple table space 535
  dropped table in segmented table space 526
  overview 525
  using pack dump backup 538
  when image copies are available 526
dropped partitions 504
DROPRECOVERY option
  description 168
  example of use 526, 531, 535, 538
  restrictions 164
DSNICOPY with INCOPY 48
DSN1LOGP utility
  example use 526
  output example 535
  using to find LRSN 535
DSN1PRNT utility 526, 531
DSNAME option 145, 182
DSNHDECP for ssid 314
DSNUM option 172, 177, 220, 258, 262, 278, 296
DSNUTILB load module 70
DSSIZE option 182
DST adjustments 594
DYNALOC installation option 601
DYNALOC option, RETRY parameter 603
dynamic allocation
building a nonpartitioned index 402
creating a consistent database for query and review 544
creating a consistent set of copies at a point in time 569
creating an image copy without reading a space 571
creating and dynamically allocating output copies 397
drop recovery using pack dump backup 538
drop recovery when image copies are available 526
dropped table in a segmented table space 531
extracting nonpartitioned keys 402
moving a table space to another subsystem 550, 558
moving an application to another subsystem 550
NGT Recover jobs 367
OUTCOPY ONLY 569, 571
overriding installation options 415
parallel index rebuilds 426, 434
QUIESCE 544
recovering a dropped table 535
recovering a dropped table space 526
recovering to a non-DB2 data set 404
recovering to a specified copy 397
recovering to a specified LRSN 400
recovering to current state 378
recovering using a change accumulation file 422
recovery plan preview 373
using LOGSCAN command 423
using SIMULATE YES option 426
using tape-stacked data sets 403
using the MAINT parameter 423
using the MAXKSOR T option 426, 434
examples, simple JCL for NGT Recover execution 311
EXCLUDE option
for the COPY command 177, 220
for the IMPORT command 295, 301
for the MIGRATE command 278, 281
excluding
index spaces 177, 220
RECOVERY MANAGER groups 177, 220, 278, 281, 295, 301
table spaces 177, 220, 278, 281, 295, 301
EXEC statement
in NGT Recover JCL 312
utility parameters 313
execution module (AFRMAIN) 335
execution phases
descriptions 358
for table space and index recovery 364, 365
for table space recovery 359, 361, 362
using Instant Snapshots 365
with BACKOUT 365
execution plan 590
EXPDT option 150
expiration date for tape copies 150
extended address volumes 65, 148, 193, 305, 582
extended format data
SHAREOPTIONS value 473
extended format sequential data set allocation 148, 582
extracted keys files work space requirements 333
extracting index keys from non-DB2 data set 241
extracting nonpartitioned keys 402

F
fallback recovery 465
FCPPRC installation option 583
features
OBID translation 35
OUTCOPY ONLY 36
running without SYSLGRNX records 36
using non-registered input copies 36
features of NGT Recover 33
fixed disk storage requirements 474
FlashCopy image copies 121, 182, 188, 472, 586, 591
FLASHCOPY option 188
forward recovery without image copy 48
FROMLOGPOINT option 279
description 180, 258, 262
LASTARCHQ 180, 258, 279
LASTCOPY 180, 258, 279
LASTQUIESCE 180, 258, 279
LASTSHUTDOWN 180, 258, 279
restrictions 164, 275
syntax diagram 656, 675
X’logPoint’ 180, 258, 279
FROMMRBA option 279
description 180, 258, 262
LASTARCHQ 180, 258, 279
LASTCOPY 180, 258, 279
LASTQUIESCE 180, 258, 279
LASTSHUTDOWN 180, 258, 279
restrictions 164, 275
syntax diagram 656, 675
X’logPoint’ 180, 258, 279
FROMMRBA or FROMLOGPOINT specification
description 279

G
GDG model 493
GDGs in copy data set names 493
group
group authorizations table 638
group attach name 314
GROUPAUTH table 638
groups 518
GRPOPTS table 637

H
hardware compression
BMCDICT table 610
hash spaces
and the BACKOUT option 468
High-speed Structure Change
description 37
DSSIZE option description 182
MODEL option description 182
PIECESIZE option description 182
TRANSFORM option description 168
HISTORY installation option
BMCHIST table 611
HISTORY option
description of installation option 584
history relationship 298, 578
AUX option
description 117, 118
HISTRETN installation option
BMCHIST table 613
HPGOBID 526, 531
HPGRBRBA 526
ICF catalog 69
IDCAMS
   SHAREOPTIONS value 473
IDCAMS load module 70
identifying maintenance applied 314
IMAGCOPY authority 68
image copies, invalid 465
image copy
   creating 55
   recovering a partitioned table space 484
   SHRLEVEL CHANGE copy 465
IMPORT command
   dependencies and prohibitions 294
   description 86
   INDEPENDENT OUTSPACE specification
      syntax diagram 679
   option descriptions 295
   OUTCOPY specification syntax diagram 679
   syntax diagram 290, 292, 679
   USING specification syntax diagram 679
IMPORT command option
   CHANGE ... FROM ...TO 276, 295
   CLONE 306
   DSNUM 296
   EXCLUDE 295, 301
   INDEPENDENT OUTSPACE 303
   INDEXES 297
   NOCOPYPEND 298
   OUTCOPY 305
   RESET 298
   STATS 298
   SYNC 297
   TABLESPACE 298
   UPDATE VERSIONS 310
   USING specification 295, 298
IMPORT command options
   ANALYZE 295
   MODEL 303
   OBJECTSET 301
   REDEFINE 295
   UPDATE VERSIONS 295
IMPORT option 520
including comments 87
INCOPY option 525, 526, 541, 550, 558
   description 188
   object ID translation 35
   restrictions 164
INCOPY specification
   description 180, 182
   syntax diagram 656
INCR option 188
   incremental copies of nonpartitioned indexes 59
   incremental image copies, cumulative 78
   INDEP INTABLESPACE option 404, 541
   INDEP OUTSPACE option 405, 531, 541, 544
   restrictions 164
   INDEPENDENT INTABLESPACE option 225, 241
   INDEPENDENT OUTSPACE option
      description 185, 223, 248
      description for IMPORT command 303
   INDEPENDENT OUTSPACE specification
      description 282
      syntax diagram 675, 679
   INDEVT option 182
   index backups
      incremental copies of nonpartitioned indexes 59
   index builds, multitasking 319, 326
   index key sort concurrency 134, 479
   index keys, extracting from non-DB2 data set 241
   INDEX option 172, 175, 219, 262, 263
   index rebuild 50
   index sort work space requirements 330
   index space status
      actions for access to index space 71
      initial 71
      required initial status 70
      unacceptable values 71
   index specification
      multiple indexes 175
      multiple indexes for LOGSCAN 263
      single index 172
      single index for LOGSCAN 262
      syntax diagram 656, 672
   INDEX(ALL) option
      restrictions 164
   INDEXES NO option 277
   INDEXES option 168, 258, 277, 297
   INDEXES RECOVER option 277
   indexes, compressed 517
   INDEXLOG option
      description for OPTION command option 124
      description of installation option 584
      restart consideration with AUTO value 339
INDEXSPACE option 172, 175, 262, 263
inflight resolution recovery 435
initial status restoration 74
inline copies
  performance 459
  restrictions 41, 193
INLINE option 188
INLOG option
  description 188
  restrictions 164
INLOG specification
  syntax diagram 656
input copies, non-registered 36
input image copy data sets 323
INSEQNO option 182
inserted partitions 504
installation option
  AUX 578
  BACKOUT 580
  CPUMIPS 581
  DATACLAS 581
  DEFINE 582
  DISKIORATE 582
  FCPPRC 583
  HISTORY 584
  KEYDSNAM 507
  KSORTSHARE 586
  MAXLSORT 588
  SNAP 592
  STORCLAS 594
  USELOGS 597
installation options
  $C30DOPT member 573
  affecting restart 339
  alphabetical listing 574
  AMSCAT 578
  AUTOSIZE 578
  BINDQUALIFIER 580
  BMCHIST 611
  CHECKINT 339, 581
  CHECKPT 339, 581
  DATAMVR 582
  EATTR 582
  ERRCONT 583
  HISTORY 611
  HISTRETN 613
  INDEXLOG 584
  IXRECP 585
  KEYDSNAM 586
  LOCCASEL 586
  LOCCPSEL 586
  MAXDRIVE 587
  MAXKSORT 587
  MAXLOGS 588
  OPNDB2ID 69, 589
  OUTCOPY 590
  PLANRECV 590
  PUBLICPLAN 590
  RCLTSK 590
  RDB2STAT 74, 590
  REMCPSEL 591
  RESINV 591
  RICHK 591
  SMCORE 591
  SORTDEVT 593
  SORTNUM 593
  TBUFFS 594
  TZRULE 594
  USEHDROBIDS 596
  WKUNIT 598
  WTOR 598
  XBMID 598
  ZIIP 599
installation options module 573
installation options module (AFROPTS) parameter in JCL 318
installation options, overriding 415
installation SYSADM authority 68
Installation System 573
Instant Snapshots
general 49, 469
migration 471
OBIDXLAT 471
registration 470
restrictions 193, 471
SNAP phase 470
with other BMC Software utilities 471
invalid image copies 465
INVOLUME option 182
issuing a QUIESCE 544
IXEXPAND option 517
IXRECP option 126, 416, 585

J

JCL
checkpoint override parameter 316
DD statements 319
EXEC statement 312
group attach name 314
input image copy data sets 323
installation options module (AFROPTS) 318
JOB statement 312
JOBLIB statement 319
message level parameter (MSGLEVEL) 316
output image copy data sets 323
RDB2STAT override parameter 317
REGION parameter 312
simple example for NGT Recover execution 311
STEPLIB DD statement 319
subsystem ID (SSID) 314
utility identifier (UTILID) 314
job restart 79
JOB statement
in NGT Recover JCL 312
parameters 312
JOBLIB statement 319
jobs
building 311
displaying status 340
restarting 335
starting 334
terminating 314

K

key data set for encrypted copies 507, 586
contents 509
management 510
overview 507
requirements 507
key sorts, shared 137
key work data sets option restrictions 91
KEYDSNAM installation option 507, 586
keys sort work files 134, 136, 326, 587, 588
KEYSORT option 129
KSORTSHARE option
description 137
installation option description 586

L

L001WK DD statements 108
L001WKnn 329, 333
large object recovery 56
last copy
migrating from 279
recovering from 180
scanning from 258
last shutdown
migrating to 284
recovering to 204
scanning to 266
LASTARCHQ option 180, 204, 258, 266, 279, 284
LASTCOMMONQ option 204, 266, 284
LASTCOPY option 180, 258, 279, 287, 398
LASTQUIESCE option 180, 204, 258, 266, 279, 284, 400
LASTSHUTDOWN option 180, 204, 258, 266, 279,
284, 550, 558
limit keys, changed 504
limitations of NGT Recover 41
LOAD LOG YES 188
LOAD LOG YES with REPLACE 36
LOB data
BMCSYNC table considerations 617
LOBs
and the BACKOUT option 122, 468
exception status and LOBs 57
general recovery 56
point-in-time recovery 56
recovery of LOG NO auxiliary spaces 56
local site copies, primary and backup 193
LOCALSEL option 211
LOCCASEL option 586
LOCSCPSEL option 586
log files, reading concurrently 490
log input phase 358
log marks, migrating to 284
log marks, recovering to 140, 204
Log Master for DB2 526, 531, 535, 569
log points
  migrating from 279
  migrating to 284
  recovering from 180
  recovering to 204
  scanning from 258
  scanning to 266
log sort space requirements 333
LOGAPPLY ONLY option 211, 270, 288
logging attribute 122, 468
LOGONLY option 212, 270, 288, 538
LOGOUnnn 329, 588
LOGPOINT LOGMARK logMarkName option 140
LOGPOINT X'logpoint' option 141
logs, maximum 588
LOGSCAN command
  dependencies and prohibitions 257
  description 86
  example 423
  index space specification 262, 263
  overview 44
  report contents 44
  syntax diagram 253
  syntax diagrams 672
  SYSSCAN 44
  table space specification 258
LOGSCAN command option
  AUX 258
  BACKOUT 266
  CLONE 265
  DSNUM 258, 262
  FROMLOGPOINT 258, 262
  FROMRBA 258, 262
  INDEX 262
  INDEXSPACE 262, 263
  LOGAPPLY ONLY 270
  LOGONLY 270
  TOLOGPOINT 266
  TORBA 266
LOGSCAN command options
  AUX 263
  INDEX 263
  INDEX (ALL) TABLESPACE 263
LOGSORT
  advantages 460
  strategy 458
LOGSORT option
  command option description 108, 188
  syntax diagram 656
LOGSORT specification
  MIGRATE command 288
  syntax diagram 675
long names in SYSIN 87
LRSN, finding with DSN1LOGP utility 531
LxxxWKnn 329, 333

M
MAINT parameter 423
MAINT restart parameter used in JCL 314
maintaining common utility tables 607
maintenance tracking 314
materialized pending changes 505
MAXDRIVE option
  installation option 587
MAXDRIVES option
  description 115
  example 415
  use with MAXLOGS 490
MAXKSORT option
  description 134
  example 426, 434
  installation option description 587
  setting 479
MAXLOGS option
  command option description 115
  installation option description 588
  usage rules 490
  use with MAXDRIVES 490
MAXLSORT option
  description 136
  installation option description 588
  setting 481
MAXPRIM option 148
MEMLIMIT parameter 66, 79, 312
MEMLIMIT system parameter 66
merge phase
  parallel 136
message level parameter (MSGLEVEL), in JCL 316
MGMTCLAS option 146
MIGRATE command
description 86
FROMLOGPOINT syntax 675
FROMRBA or FROMLOGPOINT specification syntax diagram 675
FROMRBA syntax 675
INDEPENDENT OUTSPACE specification syntax diagram 675
limitations 275
LOGSORT specification syntax diagram 675
OBJECTSET specification syntax diagram 675
option descriptions 276
syntax diagram 270, 272
syntax diagrams 675
TORBA or TOLOGPOINT specification syntax diagram 675
USING specification syntax diagram 675
MIGRATE command option
AUX 298
CLONE 288
DSNUM 278
EXCLUDE 278, 281
FROMLOGPOINT 279
FROMRBA 279
FROMRBA or FROMLOGPOINT specification 279
INDEPENDENT OUTSPACE 282
INDEXES 277
INDEXES NO 277
INDEXES RECOVER 277
LOGAPPLY ONLY 288
LOGONLY 288
LOGSORT 288
MODEL 282
OBJECTSET 281
OBJECTSET specification 281
RESET 278
STATS 278
TABLESPACE 279
TOCOPY LASTCOPY 287
TOLOCATION 290
TOLOGPOINT 284
TORBA 284
USING dataset 276
USING specification 276
MIGRATE command options
ANALYZE 276
MIGRATE command
dependencies and prohibitions 275
migrating
to a log mark 284
migrating data 544, 558
migration
from a log point 279
from the last archive quiesce point 279
from the last copy 279
from the last image copy 279
from the last quiesce point 279
from the last shutdown 279
non-DB2 data set 282
to a log point 284
to archive quiesce points 284
to last shutdown 284
to quiesce points 284
migration of copies 520
MODEL option
data set symbolic variables 185, 248
for extracting indexing keys 241
for INDEPENDENT OUTSPACE specification 185, 282, 303
for rebuilding indexes 223, 225, 248
MODELDCB option 146
moving an application to another subsystem 550
MSGLEVEL option 319
multiple commands, using 88
multiple data sets
BACKOUT 204
DSNUM ALL 172, 262
forward recovery of index spaces 500
INDEP OUTSPACE 185, 223, 248, 282, 303
INDEXLOG AUTO 339
Instant Snapshots 470, 471
MAXSORT 134, 587
OUTCOPY 193, 305
REDEFINE 80, 212, 233, 252, 308
using copies 485
multiple index specification 175, 263
multiple NGT RECOVER commands
restrictions for nonpartitioned table spaces 91
restrictions for partitioned table spaces 91
multitasking index builds 319, 326

N

names of common utility tables, determining 608
names, long 87
naming conventions
common repository 633
NEW restart parameter 314
NEW/RESTART restart parameter 314
NEW/RESTART(PHASE) restart parameter 314
NGT Copy
  utility integration 40
NGT COPY
  long names in statements 87
NGT Recover commands
  general restrictions for using multiple commands 89
  restrictions for STOGROUP-defined spaces 90
NGT RECOVER commands
  using multiple commands in a single job 88
NOCOPYPEND option 188, 234, 298
NodeID 516
NOEARLYCAT option 108
NOEARLYRECALL option 107
non-DB2 data sets
  for extracting index keys 241
  for import 303
  for migration 282
  for rebuilding indexes 223, 225, 248
  for recovery 185
non-registered copy specification, syntax diagram 656
non-registered input copies 36
nonpartitioned index recovery
  restrictions 91
  single and multiple jobs 455
nonpartitioned index, building 402
nonpartitioned indexes, incremental copies 59
nonpartitioned keys, extracting 402
NOSCRATCH option, for REDEFINE 212, 233, 252, 308
NOSYSLGRNG option 111, 525
NOT LOGGED attribute 122, 525
NOT LOGGED spaces
  and the BACKOUT option 468
NOWORKDDN option 227, 378
NOWORKDDN strategy 463
NUMREC option 110, 230, 244

example use 531
finding for dropped objects 526
finding the value for 531
for newly created objects 526
representation in PGSOBD 526
translation example 531
using the default for the source 550
using the default for the target 558
OBID translation, description 35
OBIDs, use of those in header 119, 596
OBIDXLAT option 525, 526, 531, 535, 538, 541, 544, 550, 558
description 191
restrictions 164
syntax diagram 656
object identifiers, obtaining 526, 531, 544, 550, 558
OBJECTSET option 172, 175, 177, 258, 262, 263, 265, 518
restrictions 164
OBJECTSET specification
description 177, 265, 281
for IMPORT command 301
syntax rules 177, 265
OBJECTSET specification syntax diagram 675
OBJECTSET specification
  IMPORT command 292, 679
  MIGRATE command 272
OBJSET_DEF table 634
OBJSET_SQL table 637
OBJSETS table 633
obtaining data definitions 526, 531, 535, 538
ON ERROR option, description 134
Online Consistent Copies
  and TRANSFORM 37
  as additional resource 40
  cabinet copy support 60
  DATAMVR option 129
  feature of Recovery Management solution 38
Online Consistent Copy
  and compressed indexes 517
  and FlashCopy 472
  and INCOPY 188
  and log apply 188
  and migration 541
  DATAMVR option 61
  using 61
online schema evolution 502
online schema evolution support 502
operating environment 65
operating system requirements 65
OPNDB2ID installation option 69, 589

option

ANALYZE 130, 209, 232, 245, 251, 289, 307
ARCHIVE 119, 300
AUTOSIZE 138
AUX 168, 175, 223, 258, 263, 298, 578
AVGRECSZ 110
BACKOUT 122, 204, 266, 580
BACKOUT AUTO 122
BUILDINDEX 248
CATLG 146
CHANGE ... FROM ... TO 276, 295
CHECKINT 112
CLONE 210, 227, 242, 250, 265, 288, 306
COMPRESSED 182
CPUMIPS 581
DATACLAS 146, 581
DATAMVR 129
DEFINE 121, 210, 230, 301, 582
DISKIORATE 582
DROPRECOVERY 168
DSNAME 145, 182
DSNUM 172, 177, 220, 258, 262, 278, 296
DSSIZE 182
EARLYCAT 108
EARLYRECALL 107
EATTR 148, 582
ENCIPHER 507
ENCRYPTED 182
EXCLUDE 177, 220, 278, 281, 295, 301
EXPDT 150
FCPPRC 583
FLASHCOPY 188
FROMLOGPOINT 180, 258, 262, 279
FROMRBA 180, 258, 262, 279
HISTORY 118, 584
INCOPY 180, 182, 188
INCR 188
INDEPENDENT INTABLESPACE 225, 241
INDEPENDENT OUTSPACE 185, 223, 248, 282, 303
INDEVT 182
INDEX 172, 175, 219, 262, 263
index space specification 172, 175, 262, 263
INDEXES 168, 258, 277, 297
INDEXES NO 277
INDEXES RECOVER 277
INDEXLOG 124
INDEXSPACE 172, 175, 262, 263
INLINE 188
INLOG LOGPOINT 188
INLOG RBA 188
INDEX

INSEQNO 182
INVOLUME 182
IXRECP 126
KEYSORT 129
KSORTSHARE 137, 586
LOCALSITE 211
LOGAPPLY ONLY 211, 270, 288
LOGONLY 212, 270, 288
LOGPOINT LOGMARK logMarkName 140
LOGPOINT X'logpoint' 141
LOGSORT 108, 188, 288
MAXDRIVES 115
MAXKSORT 134, 479
MAXLOGS 115
MAXLSORT 136, 481, 588
MAXPRIM 148
MGMTCLAS 146
MODEL 282, 303
MODELDCB 146
NOCOPYPEND 188, 234, 298
NOEARLYCAT 108
NOEARLYRECALL 107
NOSYSLGRNG 111
NOWORKDDN 227
NUMREC 110, 230, 244
OBIDXLAT 191
OBJECTSET 172, 175, 177, 258, 262, 265, 281, 301
ON ERROR 134
OUTCOPY 116, 193, 305
OUTCOPYDDN 198
OUTCOPYDSN 198
PART 226, 240
PIECESIZE 182
RBALRSN_CONVERSION 237
REBUILDSCOPE 121
RECOVERSCOPE 120
RECOVERYDDN 199
RECOVERYDSN 199
RECOVERYPOINT 138
RECOVERYSITE 211
REDEFINE 212, 233, 252, 308
REGISTER 193
RESET 278, 298
RESETRTS 193
RESINV 113
RESOURCE SELECTION 121
RETPD 150
REUSE 213, 234, 253, 310
RICHK 116, 168, 301
SCOPE 200
SIMULATE YES 52, 132
SKEYDDN 242, 251, 327, 328
SMCORE 113
SNAP 141, 592
SNAPSHOT 188
SORTDEVT 109, 130, 229, 243
SORTDIAG 113
SORTDYN 113
SORTNUM 109, 130, 230, 243
STACK 150
STATS 278, 298
STOGROUP 112
STORCLAS 146, 594
SYNC 297
table space specification 168, 223, 240, 250, 258
TABLESPACE 168, 223, 240, 250, 258, 279, 298
TIMESTAMP 141
TIMESTAMP with ENCRYPTED 182
TOCOPY 202
TOCOPY LASTCOPY for MIGRATE 287
TOCOPY specification 202
TOLOCATION for MIGRATE 290
TRTCH 126, 150
UNIT 146
UNITCNT 146
UNLOADKEYS 239
UPDATE VERSIONS 213, 310
URIDDDN 142
USEACCUM 111
USEHDROBIDS 119
USELOGS 597
USEORDER 112
USING 276, 295, 298
USING dataset 276
VOLCNT 146
VOLUMES 148
WORKDDN 227
WTOR 133
XBMID 127
ZIIP 128
OPTION command option

Index 697
AVGRECSZ 110
CHECKINT 112
EARLYCAT 108
INDEXLOG 124
LOGPOINT LOGMARK logMarkName 140
LOGPOINT X'logpoint' 141
LOGSORT 108
MAXKSORT 479
MAXLSORT 481
NOEARLYCAT 108
RBALRSN_CONVERSION 237
REBUILDSCOPE 121
RECOVERSCOPE 120
RECOVERYPOINT 138
RESINV 113
RESOURCE SELECTION 121
RICHK 116
SIMULATE YES 132
SMCORE 113
SNAP 141
SORTDEVT 229, 243
SORTDIAG 113
SORTDYN 113
SORTNUM 230, 243
STOGROUP 112
TIMESTAMP 141
TRTCH 126
URIDDDN 142
USEACCUM 111
USEHDROBIDS 119
USEORDER 112
WTOR 133
XBMID 127
ZIIP 128

OPTION command options

ANALYZE 130
AUTOSIZE 138
AUX 117
BACKOUT AUTO 122
DATAMVR 129
EARLYRECALL 107
HISTORY 118
IXRECP 126
KEYSORT 129
KSORTSHARE 137
MAXDRIVES 115
MAXKSORT 134
MAXLOGS 115
MAXLSORT 136
NOEARLYRECALL 107
NOSYSLGRNG 111
NUMREC 110
ON ERROR 134
SIMULATE YES 52
SORTDEVT 109, 130
SORTNUM 109, 130

options
alphabetical listing 92, 639
AUX 117
index space specification 175
OBJECTSET 263
RICHK 591
SPACE 148, 193
TAG 193

OPTIONS command 415
description 83, 103
syntax diagrams 650

OPTIONS command option
ARCHIVE 119
BACKOUT 122
OUTCOPY 116

OS requirements 65
OUTCOPY ONLY
description 36
restrictions 193

OUTCOPY option 397, 526, 538, 550, 558, 569, 571
description 116, 193
for RECOVER TABLESPACE 193
installation option description 590

OUTCOPY specification
description for IMPORT command 305
syntax diagram 679
OUTCOPY specification syntax diagram 656
OUTCOPYDDN option
description 198
data set name 145
default volume list 148
descriptor name 145
default volume list 148
descriptor name 145
expiration date 150
default volume list 148
descriptor name 145
expiration date 150
model DCB 146
number of volumes 146
operating system catalog directive 146
options for disk and tape 146
options for disk data sets only 148
options for tape data sets only 150
output allocation units 148, 193
output device name 146
retention period 150
SMS data class name 146, 581
SMS management class name 146
SMS storage class name 146, 594
stacking copies to tape 150
output files, dynamic sizing 138
output image copy data sets 323
output, controlling 319
overhead, controlling checkpoints 316
overriding installation option values 415
overview
audit controls 34
of NGT RECOVER 31
operations 38
reduced elapsed time 33
testing recovery procedures 37

parallel index key sort 134, 479
parallel index key sort, diagram 361, 362
parallel index rebuilds 361, 362, 426, 434
parallel index sorts 361, 362
parallel MERGE 481
parallel MERGE phases 136
parameter used in JCL
checkpoint override parameter 316
DB2 group attach name (SSID) 314
DB2 subsystem identifier (SSID) 314
installation options module (AFROPTS) 318
message level (MSGLEVEL) 316
RDB2STAT 317
restart parameter 314
utility identifier (UTILID) 314
parameters
MEMLIMIT 66
system 66
PART option 226, 240
partitioned index recovery
  in multiple jobs 453
  restrictions 91
partitioned table space considerations 451
partitions
  added 503
  dropped 504
  inserted 504
  rotated 503
PENDING
  SCOPE option 201, 222
performance considerations 451
PGSOBD 526
 phases
  descriptions 358
  of NGT Recover 335
  restarts allowed 335
  SNAP 470
PIECESIZE option 182
PIT_RBA 461
PLANRECV installation option 590
point of consistency, obtaining 550, 558, 569
point-in-time recovery specification
  description 204, 266
  strategies 466
previewing recovery activity
  discussion 51
  example 373
  option description 245, 251
primary copies, local and recovery site 193
processing, fallback 465
PRODREG table 638
PSID 526, 531, 544, 550, 558
PSRCP status 70
PUBLIC privilege 590
publications, related 15
PUBLICPLAN installation option 590

R

R+/CHANGE ACCUM
  change accumulation file, using 422, 488
  utility integration 41
RACF (IBM Resource Access Control Facility)
  security exit 68
RACF authority 69
RACF ID 69
RBA, finding with DSN1LOGP utility 531
RBDP index 416
RCLTSK option 590
RDB2STAT option
  general 74
  installation option description 590
RDB2STAT override parameter
  in JCL 317
  RDB2STAT(NO) 317
  RDB2STAT(RO) 317
  RDB2STAT(RW) 317
  RDB2STAT(YES) 317
RDB2STAT(NO) override parameter 317
RDB2STAT(RO) override parameter 317
RDB2STAT(RW) override parameter 317
RDB2STAT(YES) override parameter 317
real-time statistics (RTS), resetting for OUTCOPY 193
reallocated
  data sets 212, 308
  STOGROUP-defined data sets 212, 233, 252, 308
  VCAT-defined data sets 212, 308
  VSAM data sets 212, 233, 308
REBUILD INDEX command
  description 84
  option descriptions 215
  restrictions on use 89
  syntax diagram 667
  use 558
REBUILD INDEX command option

Q

QUIESCE 544, 550, 558, 569
quiesce points
rebuilding indexes by partition 219, 226
REBUILDSCOPE option 121
RECOVER BUILDINDEX command
description 86
example 402
option descriptions 246
restrictions on use 89
RECOVER BUILDINDEX command option
BUILDINDEX 248
CLONE 210
INDEPENDENT OUTSPACE 248
REDEFINE 252
SKEYDDN 251, 328
TABLESPACE 250
RECOVER BUILDINDEX command options
ANALYZE 251
CLONE 247
RECOVER INDEX command
AUX option 175
dependencies and prohibitions 164, 165, 275
description 84
index space specification 172, 175
restrictions on use 164
syntax diagrams 152, 656, 667
use 544, 550
RECOVER INDEX command option
ANALYZE 232
AUX 223
CLONE 227
INDEPENDENT INTABLESPACE 225
INDEPENDENT OUTSPACE 223
INDEX 219
NOCOPYPEND 234
NOWORKDDN 227
NUMREC 230
PART 226
REDEFINE 233
SORTDEVT 229
SORTNUM 229
TABLESPACE 223
WORKDDN 227
rebuilding indexes
by partition 219, 226
REBUILDSCOPE option 121
RECOVER BUILDINDEX command
description 86
example 402
option descriptions 246
restrictions on use 89
RECOVER BUILDINDEX command option
BUILDINDEX 248
CLONE 210
INDEPENDENT OUTSPACE 248
REDEFINE 252
SKEYDDN 251, 328
TABLESPACE 250
RECOVER BUILDINDEX command options
ANALYZE 251
CLONE 247
RECOVER INDEX command
AUX option 175
dependencies and prohibitions 164, 165, 275
description 84
index space specification 172, 175
restrictions on use 164
syntax diagrams 152, 656, 667
use 544, 550
RECOVER INDEX command option
ANALYZE 232
AUX 223
CLONE 227
INDEPENDENT INTABLESPACE 225
INDEPENDENT OUTSPACE 223
INDEX 219
NOCOPYPEND 234
NOWORKDDN 227
NUMREC 230
PART 226
REDEFINE 233
SORTDEVT 229
SORTNUM 229
TABLESPACE 223
WORKDDN 227
rebuilding indexes by partition 219, 226
REBUILDSCOPE option 121
RECOVER BUILDINDEX command
description 86
example 402
option descriptions 246
restrictions on use 89
RECOVER BUILDINDEX command option
BUILDINDEX 248
CLONE 210
INDEPENDENT OUTSPACE 248
REDEFINE 252
SKEYDDN 251, 328
TABLESPACE 250
RECOVER BUILDINDEX command options
ANALYZE 251
CLONE 247
RECOVER INDEX command
AUX option 175
dependencies and prohibitions 164, 165, 275
description 84
index space specification 172, 175
restrictions on use 164
syntax diagrams 152, 656, 667
use 544, 550
RECOVER INDEX command option
ANALYZE 232
AUX 223
CLONE 227
INDEPENDENT INTABLESPACE 225
INDEPENDENT OUTSPACE 223
INDEX 219
NOCOPYPEND 234
NOWORKDDN 227
NUMREC 230
PART 226
REDEFINE 233
SORTDEVT 229
SORTNUM 229
TABLESPACE 223
WORKDDN 227
rebuilding indexes by partition 219, 226
REBUILDSCOPE option 121
RECOVER BUILDINDEX command
description 86
example 402
option descriptions 246
restrictions on use 89
RECOVER BUILDINDEX command option
BUILDINDEX 248
CLONE 210
INDEPENDENT OUTSPACE 248
REDEFINE 252
SKEYDDN 251, 328
TABLESPACE 250
RECOVER BUILDINDEX command options
ANALYZE 251
CLONE 247
RECOVER INDEX command
AUX option 175
dependencies and prohibitions 164, 165, 275
description 84
index space specification 172, 175
restrictions on use 164
syntax diagrams 152, 656, 667
use 544, 550
RECOVER INDEX command option
ANALYZE 232
AUX 223
CLONE 227
INDEPENDENT INTABLESPACE 225
INDEPENDENT OUTSPACE 223
INDEX 219
NOCOPYPEND 234
NOWORKDDN 227
NUMREC 230
PART 226
REDEFINE 233
SORTDEVT 229
SORTNUM 229
TABLESPACE 223
WORKDDN 227
rebuilding indexes by partition 219, 226
REBUILDSCOPE option 121
RECOVER BUILDINDEX command
description 86
example 402
option descriptions 246
restrictions on use 89
RECOVER BUILDINDEX command option
BUILDINDEX 248
CLONE 210
INDEPENDENT OUTSPACE 248
REDEFINE 252
SKEYDDN 251, 328
TABLESPACE 250
RECOVER BUILDINDEX command options
ANALYZE 251
CLONE 247
RECOVER INDEX command
AUX option 175
dependencies and prohibitions 164, 165, 275
description 84
index space specification 172, 175
restrictions on use 164
syntax diagrams 152, 656, 667
use 544, 550
RECOVER INDEX command option
RECOVER INDEXSPACE command options
   ANALYZE 167
   CLONE 167
   REDEFINE 167

RECOVER OBJECTSET command option
   OBJECTSET 265

RECOVER OBJECTSET command options
   OBJECTSET 177

RECOVER TABLESPACE command
dependencies and prohibitions 164, 275
   description 84
   restrictions on use 164
   syntax diagrams 656
   table space specification 168

RECOVER TABLESPACE command options
   ANALYZE 167
   CLONE 167
   DSNUM 177
   FROMLOGPOINT 180
   REDEFINE 167
   UPDATE VERSIONS 167
RECOVER UNLOADKEYS command
  description 85
  option descriptions 237
  syntax diagram 670
RECOVER UNLOADKEYS command option
  ANALYZE 245
  INDEPENDENT INTABLESPACE 241
  NUMREC 244
  PART 240
  SKEYDDN 242, 327
  SORTDEV 243
  SORTNUM 243
  TABLESPACE 240
  UNLOADKEYS 239
RECOVER UNLOADKEYS command options
  CLONE 238
RECOVERDB authority 68
recovering
  dropped table from a simple table space 535
  dropped table in segmented table space 531
  dropped table space, description 525
  to a non-DB2 data set 404
  to a specified copy 397
  to a specified LRSN 400
  to the current state 378
RECOVERSCOPE option 120
recovery
  after added partitions 503
  after limit key changes 504
  change accumulation file, using 488
  dropped partitions 504
  from a log point 180
  from the last archive quiesce point 180
  from the last copy 180
  from the last image copy 180
  from the last quiesce point 180
  from the last shutdown 180
  hash spaces 468
  inserted partitions 504
  LOBs 56, 468
  multiple indexes 477
  multiple partitions concurrently 451
  nonpartitioned indexes 91
  NOT LOGGED spaces 468
  output 405
  partitioned indexes 91
  plan 373
  rotated partitions 503
  simulation 426
  to a log mark 140, 204
  to a log point 204
  to archive quiesce points 204
  to last shutdown 204
  to quiesce points 204
  using a full volume backup 77
  using BUILDINDEX 50
  using incremental image copies 78
  using NOWORKDDN 463
  using only the log 48
  using SHRLEVEL CHANGE copies 464
  using UNLOADKEYS using 50
  using UNLOADKEYS/BUILDINDEX 463
  while maintaining activity while 54
  without an image copy 48
Recovery Management for DB2 32, 37, 138
RECOVERY MANAGER for DB2 569
RECOVERY MANAGER groups 518
RECOVERY MANAGER utility integration 40
recovery plan, option descriptions 130, 232, 245, 251
recovery simulation 52, 132, 167, 219, 239, 247
recovery site copies, primary and backup 193
recovery strategies
building nonpartitioned indexes on partition table space 50
compressed indexes 517
eliminating interim data set for index keys 50
fallback 465
large partitioned objects 490
large table spaces 490
LOGSORT 458
maintaining activity 54
planning recovery resources 489
previewing recovery activity 51
R+/CHANGE ACCUM, using 488
recovery with an image copy 48
recovery with an Instant Snapshot 49
resource selection 50
separate change accumulation files, using 490
separate copies, using 490
sorting the log 458
testing recovery 51
using BUILDINDEX 50
using UNLOADKEYS 50

recovery, local
BACKOUT AUTO option 47
backout recovery 47
RECOVERYDDN option
description 199
syntax diagram 656
RECOVERYDSN option, description 199
RECOVERYPOINT 57
RECOVERYPOINT option, description 138
RECOVERYSITE option 211
RECP status 416
REDEFINE option 212, 233, 252, 308
redirection of output 405
reduced elapsed time 33
REGION parameter 312
REGISTER option 193
registering index backups 58
registration in the BMCXCOPY table 54
related publications 15
REMCPSSEL option 591
removing objects for restart 335
REORG LOG YES 36, 188
REPAIR VERSIONS 513
repository tables 633
requesting jobs, restarting 79
requirements
extracted keys files work space 333
fixed disk storage 330
index sort work space 330
log sort space 333
sort work space 330, 333
virtual memory 330
RESET option 278, 298
RESETRTS option 193
residing in APF library 70
RESINV option 113, 591
RESOURCE SELECTION option 121
resource selection, using 50
restart
affected by installation options 79
at a MERGE phase 335
at beginning of LOG INPUT phase 335
at the beginning of the last incomplete phase 314
considerations when creating output copies 337
during BUILD phase 335
from last incomplete phase 314, 335
from last sync point 314
from point of failure 337
installation options that affect 339
limitations 228
LOGSORT restriction 458
NGT Recover job 335
removing/adding objects 335
syntax options that affect 335
table for NGT Recover phases 335
transform 338
using checkpoints for restart 79
using INDEXLOG AUTO 339
using sync restart 314
utility parameters that affect 314
restart parameter
MAINT 314
NEW 314
NEW/RESTART 314
NEW/RESTART(PHASE) 314
RESTART 314
RESTART(PHASE) 314
TERM 314
restoration of initial status 74, 317
restrictions
for key work data sets 91
for multiple RECOVER commands 89, 91
for NGT Recover 41
for nonpartitioned table spaces 91
for partitioned table spaces 91
for REBUILD INDEX 89
for RECOVER BUILINDEX 89
for SKEYDDN 91
for SORTDEVT 91
for SORTNUM 91
for STOGROUP-defined spaces 90
for WORKDDN 91
using LOGSORT 460
using multiple commands 89
retention period, tape copies 150
RETPD option 150
RETRY parameter of DYNALOC option 603
reverting dynamic allocation of SORTWK 601
REUSE option 213, 234, 253, 310
reusing data sets 213, 234, 253, 310
revision bars 17
RICHK option 168, 301
description of installation option 591
rotated partitions 503
RTS, resetting for OUTCOPY 193
running BMC utilities concurrently 619

S

S001WKnn 326, 329
scan
from a log point 258
from the last archive quiesce point 258
from the last copy 258
from the last image copy 258
from the last quiesce point 258
to a log point 266
to archive quiesce points 266
to last shutdown 266
to quiesce points 266
SCOPE option
ALL 201
description 200
PENDING 201, 222
STATUS 222
UPDATED 201
SCOPE option, description 200
secondary extents, sliding-scale calculation 185, 223, 248
security
mechanisms 68
setting SMCORE installation option values 477
shared key sorts 137
SHAREOPTIONS value 473
SHRLEVEL 614
SHRLEVEL CHANGE copies used for recovery 464
SHRLEVEL of copies produced 567
shutdown
migrating from 279
migrating to last 284
recovering from 180
recovering to last 204
scanning from 258
scanning to last 266
SIMRBLD 219
SIMRBLD command 426
SIMRCVR 167, 219, 239, 247
SIMRCVR command 426
SIMULATE option 52
SIMULATE YES option, description 52, 132
SIMULATE YES option, example use 426
simulating recovery 426
simulation 52
simulation mode 52, 132, 167, 219, 239, 247
SKEY DD statement 402
SKEYDDN option
description 242, 251
restart requirements 335
restrictions 91
unloaded keys file space requirements 333
sliding-scale calculation, for secondary extents 185, 223, 248
SMCORE installation option 477, 591
SMCORE option 113
SNAP installation option 592
SNAP option
description for OPTION command 141
snap phase 358
SNAP phase 470
SNAPSHOOT option 188
sort concurrency 136, 481
sort message files 134, 136, 326, 329, 587, 588
sort options 113
sort performance 474
sort sizes, determining 474
sort tasks
starting a new NGT Recover execution 314
starting a NGT Recover job 334
starting a space after recovery 74
STATS option 278, 298
status
displaying for jobs 340
index space-unacceptable 71
table space-unacceptable 71
when NGT Recover job is terminated 340
STATUS
SCOPE option 222
status, BMC utilities 608
STEPLIB DD statement 319
STOGROUP option 112
STOGROUP-defined data sets
for index spaces 69
reallocating 212, 233, 308
STOPDB authority 68
storage
fixed disk requirements 330, 474
virtual memory requirements 474
virtual, specified by REGION parameter 312
storage, virtual 66
STORCLAS installation option 594
STORCLAS option 146
strategy
LOGSORT 458
NOWORKDDN 463
UNLOADKEYS/BUILDINDEX 463
striped data
SHAREOPTIONS value 473
subsystem ID (SSID) in JCL 314
summary of product changes 17
SxxxWKnn 134, 136, 326, 329, 587, 588
symbolic variable
&PART4 496
&PART5 497
&UNIQ or &UQ 497
symbolic variables

management 359
parallel 361, 362
sort work data set
defining 329
options for REBUILD INDEX 229
options for RECOVER UNLOADKEYS 243
sort work space
allocating 483
requirements 330, 333, 475
SORTDEVT option
description 109, 130, 229, 243
example 415
general information 90
installation option description 593
restrictions 91
SORTDIAG option 113
SORTDYN option 113
sorting the log 459
SORTNUM option
description 109, 130, 230, 243
example 415
general information 90
installation option description 593
restrictions 91
SPACE option 148, 193
specification
EXEC statement 312
JOB statement 312
SPUFI 526
SQL statements
deleting rows from the BMCDICT table 610
deleting rows from the BMCHIST table 614
deleting rows from the BMCSYNC table 618, 626
deleting rows from the BMCSYNC table for RECOVER UNLOADKEYS 619
deleting rows from the BMCXCOPY table 631
displaying BMC utilities 608
querying BMCHIST table 608
querying BMCXCOPY table 608
terminating BMC utilities 608
STACK option 150
stacked tape
change accumulation file, using 490
image copies, dynamic allocations 498
mounts 323
ordering copies 485
strategies in using copies 485
START_RBA 461
STARTDB authority 68
in copy data set names 494
with IMPORT 303
with IMPORT command 303
with MIGRATE 282
with MIGRATE command 282
with OUTPUT command 145, 146
with RECOVER INDEX 185
with RECOVER INDEX command 182
with RECOVER INDEXSPACE 185
with RECOVER INDEXSPACE command 182
with RECOVER TABLESPACE 185
with RECOVER TABLESPACE command 182
SYNC option 297
syntax diagram
IMPORT command 679
syntax diagrams
FROMLOGPOINT specification 656, 675
FROMRBA specification 656, 675
IMPORT command 290, 292
INCOPY specification 656
INDEPENDENT OUTSPACE specification 656, 675, 679
index specification 152, 656, 667, 672
LOGSCAN command 253, 672
LOGSORT specification 650, 656, 675
MIGRATE command 270, 272, 675
non-registered copy or INLOG 656
OBIDXLAT specification 656
OBJECTSET specification 272, 292, 675, 679
OPTIONS command 650
OUTCOPY specification 656, 679
OUTCOPYDDN specification 292, 656
OUTPUT command 654
point-in-time recovery specification 656
REBUILD INDEX command 667
RECOVER INDEX command 656
RECOVER INDEXSPACE command 656
RECOVER TABLESPACE command 656
RECOVER UNLOADKEYS command 670
RECOVERYDDN specification 656
SORT specification 650
table space specification 656, 672
TOCOPY specification 656
TOLOGPOINT option 656, 675
TORBA option 656
TORBA or TOLOGPOINT specification 675
USING specification 675, 679
SYSDUMP 319
SYST1 DD statement
and key work data sets 228
eliminating interim data sets 50
with key work data sets 326

T

table names, determining 608
table space recovery in multiple jobs 453
table space specification syntax diagram 656, 672
table space status
actions for access to table space 71
initial 71
overview 70
unacceptable values 71
when terminating an NGT Recover job 340
table space, partitioned 451
tables
BMCSYNC 314
BMCUTIL 314
tables, BMC
backing up 607
BMCHIST 611
BMCSYNC 614
BMCTRANS 622
BMCUTIL 623
BMCXCOPY 626
considerations 607
determining names 608
querying 608
warnings 607
tables, BMC Common DB2 repository

SYSADM authority 68
SYSCOPY table 535
SYSCTRL authority 68
GROUPAUTH 638
GRPOPTS 637
OBJSET_DEF 634
OBJSET_SQL 637
OBJSETS 633
PRODREG 638

Tables, BMCDICT 610
TABLESPACE option 168, 223, 240, 250, 258, 279, 298
TAG option 193
tape copies
  expiration date 150
  retention period 150
Tape Mount Management 150
tape mounts, stacked 323
tape pick list 51
tape stacking example 403
TBUFFS option 594
TDES (Triple Data Encryption Standard) 506
temporal tables 117, 118, 298, 578
TERM restart parameter 314
terminating BMC utilities 608
termination
  NGT Recover job 340
termination, NGT Recover job 314
testing recovery strategy 37, 52
TIMESTAMP option 141, 182
timestamp recovery 138, 435
TMM (Tape Mount Management) 150
TOCOPY LASTCOPY option 287
TOCOPY option
description 202
example 397
syntax diagram 656
TOCOPY specification
description 202
  LASTCOPY option 287
  restrictions 164, 275
TOLOCATION option 290
TOLOGPOINT LOGMARK logMarkName 204, 284
TOLOGPOINT option 400, 526, 531, 544
  BACKOUT 204, 266
description 204, 266
  LASTARCHQ 204, 266, 284
  LASTCOMMONQ 204, 266, 284
  LASTQUIESCE 204, 266, 284
  LASTSHUTDOWN 204, 266, 284
  X'logPoint' 204, 284
TOLOGPOINT specification
syntax diagram 675
TORBA LOGMARK logMarkName 204, 284
TORBA option 284, 400, 544, 550, 558
  BACKOUT 204, 266
description 204, 266
  LASTARCHQ 204, 266, 284
  LASTCOMMONQ 204, 266, 284
  LASTQUIESCE 204, 266, 284
  LASTSHUTDOWN 204, 266, 284
  X'logPoint' 204, 284
TORBA or TOLOGPOINT specification
  for the MIGRATE command 284
TORBA specification
  syntax diagram 675
TORBA/TOLOGPOINT specification
  syntax diagram 656
tracking maintenance applied 314
TRANSFORM option 37, 168
transform, restarting 338
Triple Data Encryption Standard (TDES) 506
TRTCH option 126, 150
TZRULE option 594

UNIT option 146
UNITCNT option 146
unload phase 358
unloading indexes by partition 240
UNLOADKEYS entries, cleaning up 619
UNLOADKEYS option
  and indexes defined using an expression 239
description 239
UNLOADKEYS/BUILDINDEX strategy 463
UPDATE VERSIONS option 213, 310
UPDATED
  SCOPE option 201
URIDDDN option 142
USEACCUM option 111, 422
USEHDROBIDS installation option 596
USEHDROBIDS option 119
USELOGS installation option 597
USEORDER option 112
USING specification
description 276, 295, 298
using specification syntax diagram 675, 679
utility identifier (UTILID) in JCL 314
utility integration
DB2 resources 40
NGT Copy 40
overview 40
R+/CHANGE ACCUM 41
RECOVERY MANAGER 40
utility parameter
checkpoint override parameter 316
group attach name 314
installation options module (AFROPTS) 318
message level 316
on the EXEC statement 313
RDB2STAT 74
RDB2STAT override parameter 317
restart parameter 314
subsystem ID (SSID) 314
utility identifier (UTILID) 314

V
VCAT-defined data sets
option ignored (REDEFINE) 233, 252
reallocating 212, 308
version information 513
versioning relationship 117, 298, 578
versioning, steps to handle 513
virtual memory requirements 330, 474
virtual storage 66
virtual storage used by NGT Recover 312
VOL=SER parameter 337
VOLCNT option 146
VOLUMES option 148
VSAM data sets
and REDEFINE NO 233
reallocating 212, 308

W
WKUNIT installation option 598
work data set ddname 227
work data sets, defining for sorts 329
work file 402
work space requirements 333
WORKDDN option
description 227
restrictions 91
working with ASCII data 473
WTOR installation option 598
WTOR option on the OPTION command 133

X
XBM subsystem ID 127, 598
XBMID option 127, 598
XML data
BMCSYNC table considerations 617
XML objects
DocID 516
NodeID 516
recovering 516

Z
zIIP enablement 599
ZIIP installation option 599
ZIIP option
and the OPTION statement 128
and the XBMID option 128
software requirements 67
zIIP processors 66